Les virus du futur

Quand les mathématiques travaillent pour le côté obcur

Eric Filiol
filiol@esiea.fr

ESIEA Laval
Laboratoire de virologie et de cryptologie opérationnelles


Les virus
sont désorm
Les virus inconnus représente
pour votre entreprise. Les n
TruPrevent surveillent,
interceptent ces nouvel
TRUEPREVENT
G DATA
SECURITY
Technologie primée
Deux modules
anti-virus pour
une double
protection
DOUBLESCAN
OUTBREAK SHIELD
AntiVirusKit 2006
Protection à 100% contre les virus connus et inconnus !
OutbreakShield • Deux modules antivirus • Protection à 100% contre les virus, vers,
backdoors, spywares, chevaux de Troie, dialers • Mise à jour horaire
es virus attac
nt que les mises
arrivent !
son réseau
épendre
signatures !
GUARD
ises à jour
ersion en 1998 !
bloqués par défaut.
ViGUARD n'ont jamais
GUARD
ntercepté !
et connaître
es et partenaires
73 33 33 www.do
Microsoft
days
ec vous 09

Introduction

Résultats théoriques (Cohen - 1986)

« La détection des virus un un problème indécidable (I.e. sans solution) »

- Il n’existe aucun programme capable de détecter tout virus.

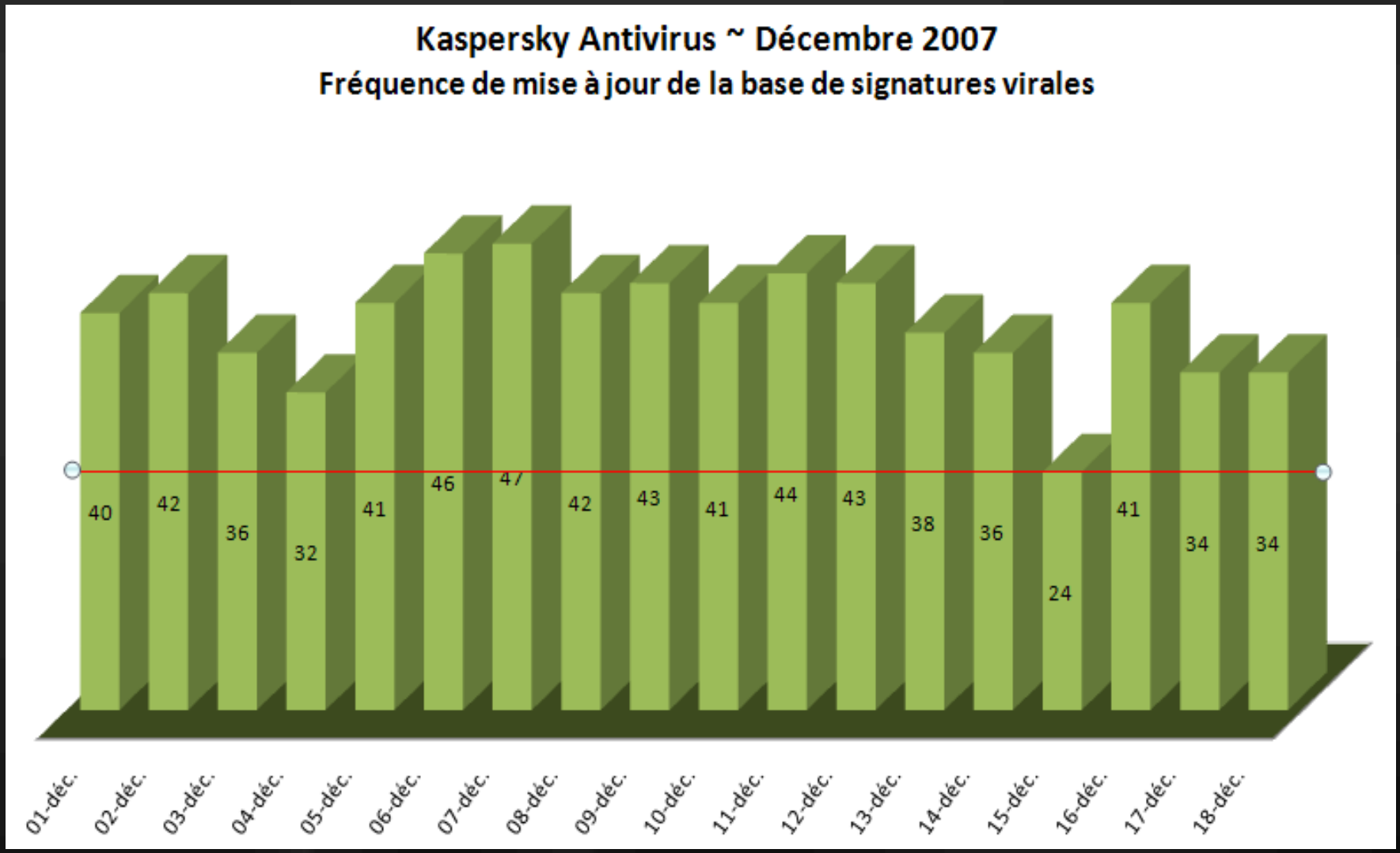
Introduction

Fait (réalité de l'attaquant)

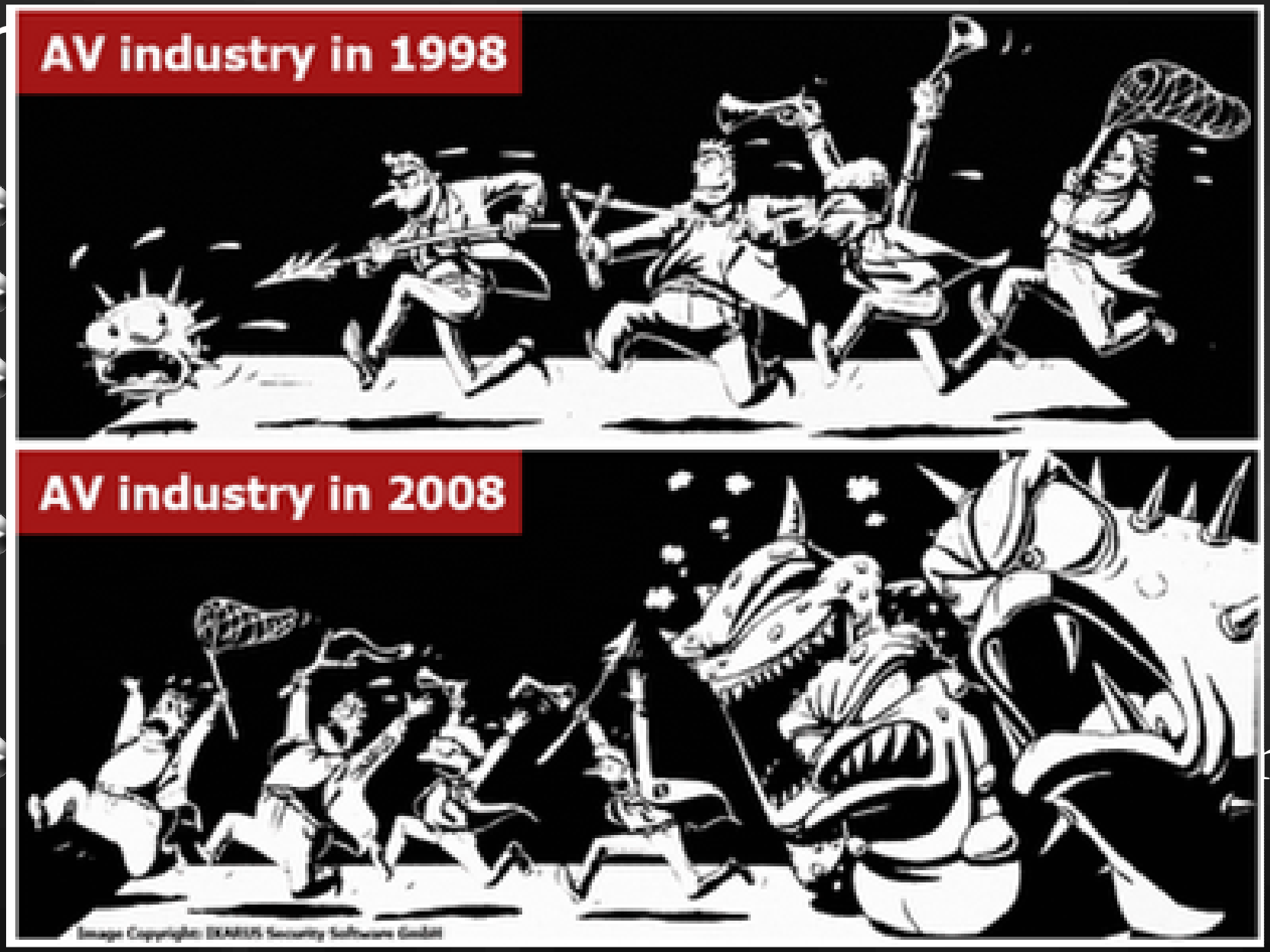
« Donnez moi un système soit-disant sécurisé ou un produit de sécurité inviolable…et je trouverai comment le contourner ou l'attaquer d'une manière ou d'une autre ».

- De nombreux exemples ces dernières années (ex. sécurité de l'iPhone).

Introduction



In



Introduction

- But de la présentation :
  - Comprendre comment la menace va très probablement évoluer.
  - Comprendre pourquoi les vendeurs d'antivirus ont échoué.
  - Comprendre quels seront les outils d'une « véritable » cyberguerre.
  - Expliquer comment organiser la prévention.

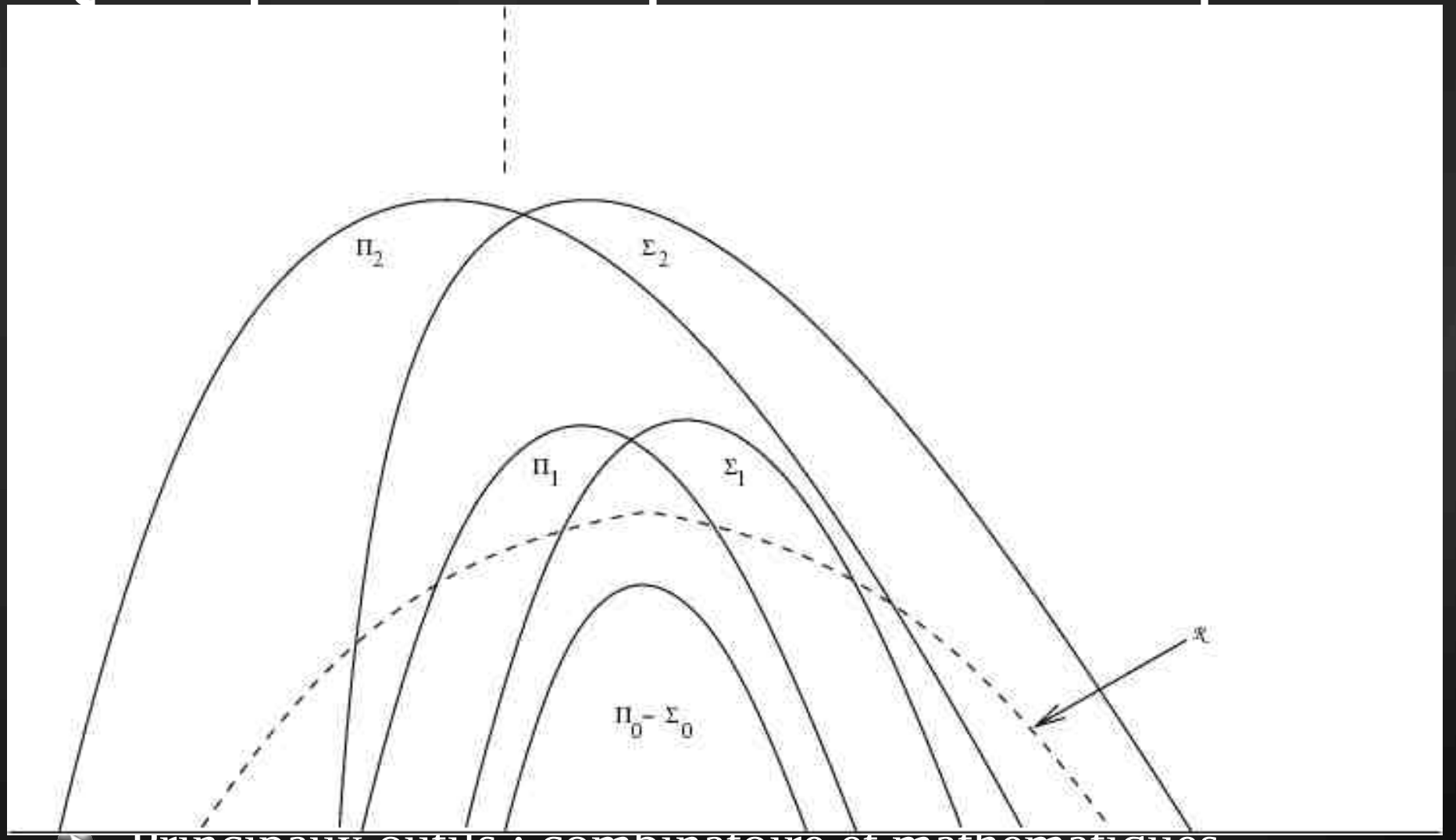
Plan

- Introduction.
- Concepts mathématiques pour les nuls !
  (désolé !)
- Principes de base de la conception de codes malveillants.
- Quelques exemples et cas.
- Conclusion.

Quelques concepts mathématiques

- **Théorie de l'information**
  - Concept central $\Rightarrow$ entropie.
  - Permet de caractériser la quantité d'information.
  - Toute source d'information peut être cqrqctérisée par son profil d'entropie (programmes, langages, données…).
  - Pour une quantité secrète, définit la quantité de secret ou d'incertitude (quantifie effort de l'attaquant).
- **Outils principaux**
  - Théorie des probabilités et des statistiques.
  - Simulabilité des tests statistiques (Filiol - 2007).
    - Dis moi quels tests tu utilises et je pourrais les leurrer et contourner la détection.
  - Cryptologie et stéganographie.

Quelques concepts mathématiques



- Principaux outils : combinatoire et mathématiques discrètes.

Quelques concepts mathématiques

- **Théorie de la calculabilité.**
  - Concept central ⇒ machine de Turing.
  - Décider s'il existe une machine de Turing (un programme) pouvant résoudre un problème donné.
  - Certains problèmes ne sont pas calculables (la machine de Turing correspondante ne s'arrête jamais).
  - En conséquence, le problème n'a pas de solution !
  - Exemple célèbre : le problème de la détection virale !
- **Outils** : grammaires et langages formels.

Principes de base de la conception de codes malveillants (indétectables).

Principes de base

- Concevoir le code de telle sorte que pour l'antivirus le problème de la détection :
  - soit « dur » à calculer (NP et au-delà),
  - soit ne soit pas calculable.
- Exploiter le fait que les AV ne sont que des produits commerciaux.
  - Un AV consacre quelques cycles pour l'analyse seulement $\Rightarrow$ le virus va en prendre plus.
    - (τ-obfuscation – Beaucamps – Filiol 2006).

Principes de base (2)

- Leurrer les algorithmes de détection.
  - Tout algorithme de détection peut être modélisé par un test statistique (Filiol – Josse 2007).
  - Utiliser des techniques de simulabilité (Filiol 2007).
  - Utiliser des techniques de cryptographie et de statistiques « malicieuses » (Filiol – Raynal CanSecWest 2008).
  - Utiliser des techniques de blindage pour interdire l'analyse de code.
    - Codes Bradley (Filiol 2005).
- Imaginer des nouvelles formes de malware.
- Et combiner tout cela !

Principes de base (3)

- Au niveau du code penser à la fois en termes de :
  - Détection par analyse de forme,
  - ET de détection comportementale.
  - Il faut contouner les deux !
  - Exemple d'échec : GpCode (2008).
- Analyser la cible (utilisateur, produit AV, environnement…).

Quelques exemples et cas…

… parmi de nombreux possibles !

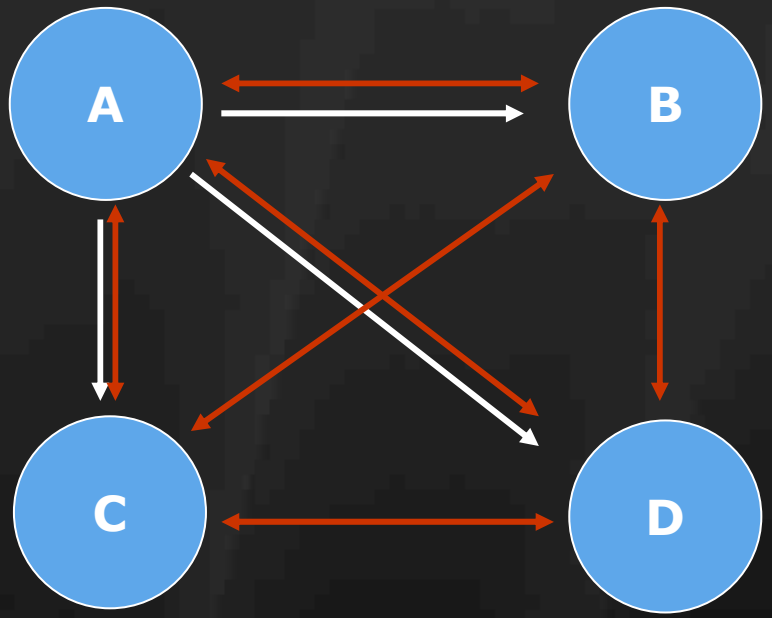
Quelques exemples et cas.

- Exemples et cas provenant :
  - Expertises judiciaires (analyse forensique).
  - Analyse d'attaques ciblées réelles.
  - Recherche et expérimentations.
- Ce qu'il FAUT garder à l'esprit :
  - Attaque réussie = Code + protocole d'attaque.
  - Considérer le code seulement est d'une utilité très limitée.
    - Pensée combinée du renseignement et de l'infanterie.

Codes viraux K-aire ou diviser l’information virale

Codes K-aire.

Idée de départ : un cas réel (2004)



- Le virus installe trois variantes de lui-même en mémoire.
- Les variantes sont des versions faiblement de A.
- Les variantes s'auto-rafraîchissent en permanence (suicide, regénération, mutation et ainsi de suite…).

Chaque fois qu'un AV réussit à tuer une des variantes, les autres la réinstallent.

Codes K-aires (formalisation - Filiol 2007)

- Définition: famille de k (non nécessairement tous exécutables) fichiers dont la réunion constitue un malware et dont l'action combinée est celle d'un malware. Chaque partie apparaît comme anodine pour les AV.
- Deux types différents :
  - Codes k-aires parallèles.
  - Codes k-aires série.
- Possibilité de combiner les deux.
  - Codes k-aires série/parallèle.

Codes K-ary (formalisation)

- Pour chaque type, trois classes distinctes :
  - Sous-classe A (parties dépendantes).
  - Sous-classe B (parties indépendantes).
  - Sous-classes C (parties faiblement dépendantes)
- Validation par différents PoC:
  - OpenOffice Virus Final_Touch (de Drézigué at al. 2006).
  - PoC_Serial (Filiol 2007) avec $4 \leq k \leq 8$ (trois sous-classes).
  - PoC_Parallel (Filiol 2007) avec $k = 4$ (trois sous-classes).
- Aucune détection par les AV.

Codes K-ary (formalisation)

- La détection des codes k-aires a été démontrée comme étant un problème NP-complet.
  - NP complet si les fonctions booléennes d'interactions sont déterministes.
- Possibilité de concevoir des codes encore plus sophistiqués :
  - Fonctions d'interaction non déterministes.
  - Utilisation de schémas combinatoires (ex schémas à seuil).
- Travaux en cours.

Propagation optimisée de vers
…où comment concevoir un botnet (quasi) parfait.

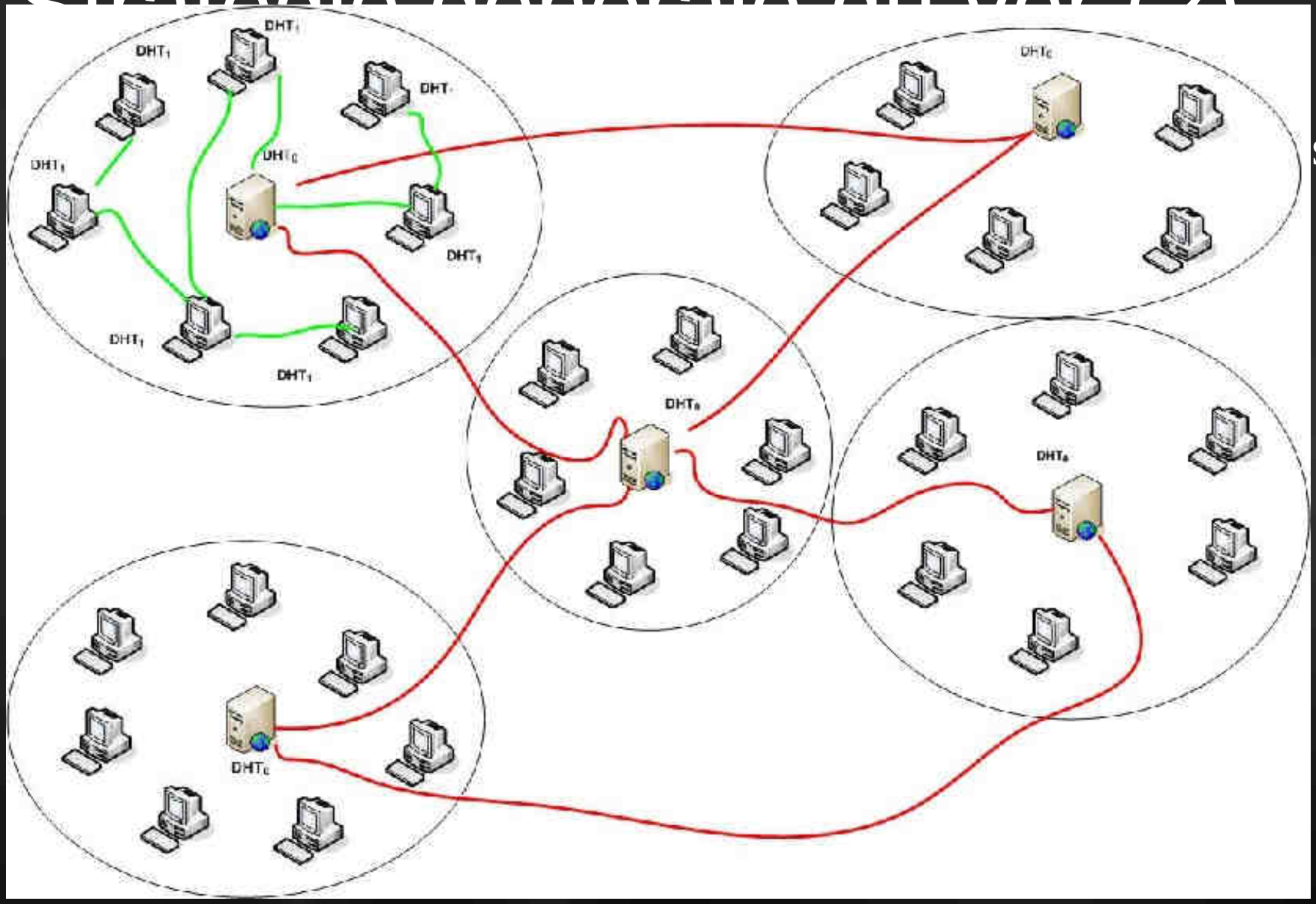
Propagation optimisée de vers

- Comment concevoir un ver furtif et rapide pour infecter un réseau de type Internet inconnu ?
  - Organiser un réseau malicieux (botnet) à deux niveaux.
  - Utiliser certaines structures combinatoire pour propager et administrer le ver.
  - Aucune connaissance du réseau nécessaire *a priori*.
- Le taux de connexions inutiles est optimalement réduit.
- PoC et simulateur SuWast (Filiol et al. 2007)

Stratégie générale du ver.

- Le réseau cible est organisé par le ver selon une hiérarchie à deux niveaux.
  - Localement, des réseaux « malicieux » de type P2P sont installés (réseaux bas ; gestion locale d'adresses dynamiques).
  - Chaque réseau malicieux bas inclut également une adresse IP fixe.
  - A un niveau plus haut, un réseau malicieux considère uniquement des adresses IP fixes (réseau malicieux haut).
  - Globalement, une structure de graphe G est utilisée pour gérer ces adresses IP fixes seules (au niveau de l'attaquant).
- L'outil de base utilisé pour gérer tous ces réseaux :
  - DHT (Dynamic Hash Tables).

Stratégie générale du ver (2)



se

de

Mécanisme de propagation.

Cette étape vise à trouver des adresses IP cibles à infecter.

1. Avec une probabilité $p_0 < 0.1$, génération d'une adresse IP aléatoire. Le ver tente d'infecter cette adresse.
2. Ensuite le ver cherche à infecter des adresses IP existantes localement :
   - Table ARP, répertoires d'applications (navigateur Internet, antivirus, firewall...).
   - Identification de machines déjà connectées à la machine locale (*netstat*, *nbtstat*, *nslookup*, *tracert* ...).
3. Tentative de se propager à ces adresses et mise à jour des structures DHT en cas de succès.
4. Informations envoyées à la machine C & C de l'attaquant.

Le ver est capable de déterminer si une cible est infectée ou non.

Données collectées.

- Pour surveiller et gérer l'activité du ver, l'attaquant utilise certains estimateurs.
- La structure de graphe G (dirigé) correspondant au réseau malicieux haut est définie comme suit :
  - Chaque adresse IP fixe est un nœud du graphe.
  - Le noeud i est connecté au nœud j si la machine j a été infectée par la machine i.

Données collectées (2).

- Supposons que la machine i a infectée la machine j à l'instant t. Les données suivantes sont alors renvoyées à la console C & C :
  - Adresse IP de la machine i .
  - Adresse IP de la machine j .
  - Une unique adresse IP fixe.
  - La donnée t.
  - La marque d'infection (la machine j était ou non déjà infectée)

Administration du réseau infecté

- Une fois que le ver a infecté toutes les machines possibles, l'attaquant doit contrôler et administrer le ver et son
- Utilisation des structures DHT se sorte à limiter la taille des données et des structures.
  - Systématiquement, les DHTs d'une machine i donnée gère dynamiquement uniquement les adresses IP des machines qui s'y sont connectées récemment.
- Utilisation d'un système d'identification de nœuds (node ID)
  - Adresse IP, données dans la machine…
  - Gestion de la distance entre nœuds par métrique XOR
  - Utilisation de la technique Kademlia.

Administration du réseau infecté (2)

- Utilisation d'une mesure pondérée pour chaque adresse IP dans les tables DHTs. Considérons la DHT^i_1 de la machine i .
  - Pour chaque autre adresse IP j dans DHT^i_1 , notons d_{ij} la distance (xor) entre les machines i et j, et t_{ij} le dernier instant de connexion (en secondes) entre les machines i et j .
  - Considérons la pondération suivante :

$$w_{ij} = d_{ij} \times t_{ij} .$$

- Ainsi, la table DHT^i_1 s'auto-rafraîchit en permanence pour conserver seulement un nombre réduit d'adresses IP de poids w_{ij} minimal.

Le graphe d'infection

- Le but est de modéliser les connexions entre les adresses IP fixes au moyen du graphe dirigé G.
  - Les nœuds de G, notés (n_i) $1 \leq i \leq N$ représentent les adresses IP fixes (généralement un serveur) ;
- Les entrées de la matrice d'incidence de G sont définies par :
  - $a_{i,j} = 1$ si la machine j a été infectée par la machine i,
  - Sinon $a_{i,j} = 0$.

Administration du réseau infecté (3)

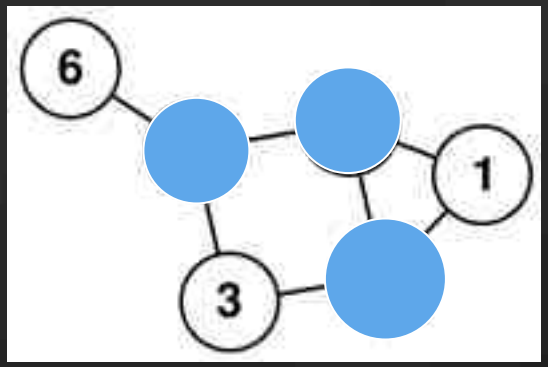
- Rechercher une structure de « vertex cover » dans le graphe.
- Définition : Soit G un graphe simple (V , E). Le vertex cover est un sous-ensemble V ′ des nœuds de V contenant au moins un des deux nœuds de chaque arête :

  $V' \subset V : \forall \{a, b\} \in E, a \in V' \text{ ou } b \in V'$

- Le problème du vertex cover est NP-complet.
- Il existe toutefois des heuristiques efficaces (Dharwadker 2006).

Administration du réseau infecté (4)

- Considérons le graphe simple suivant :



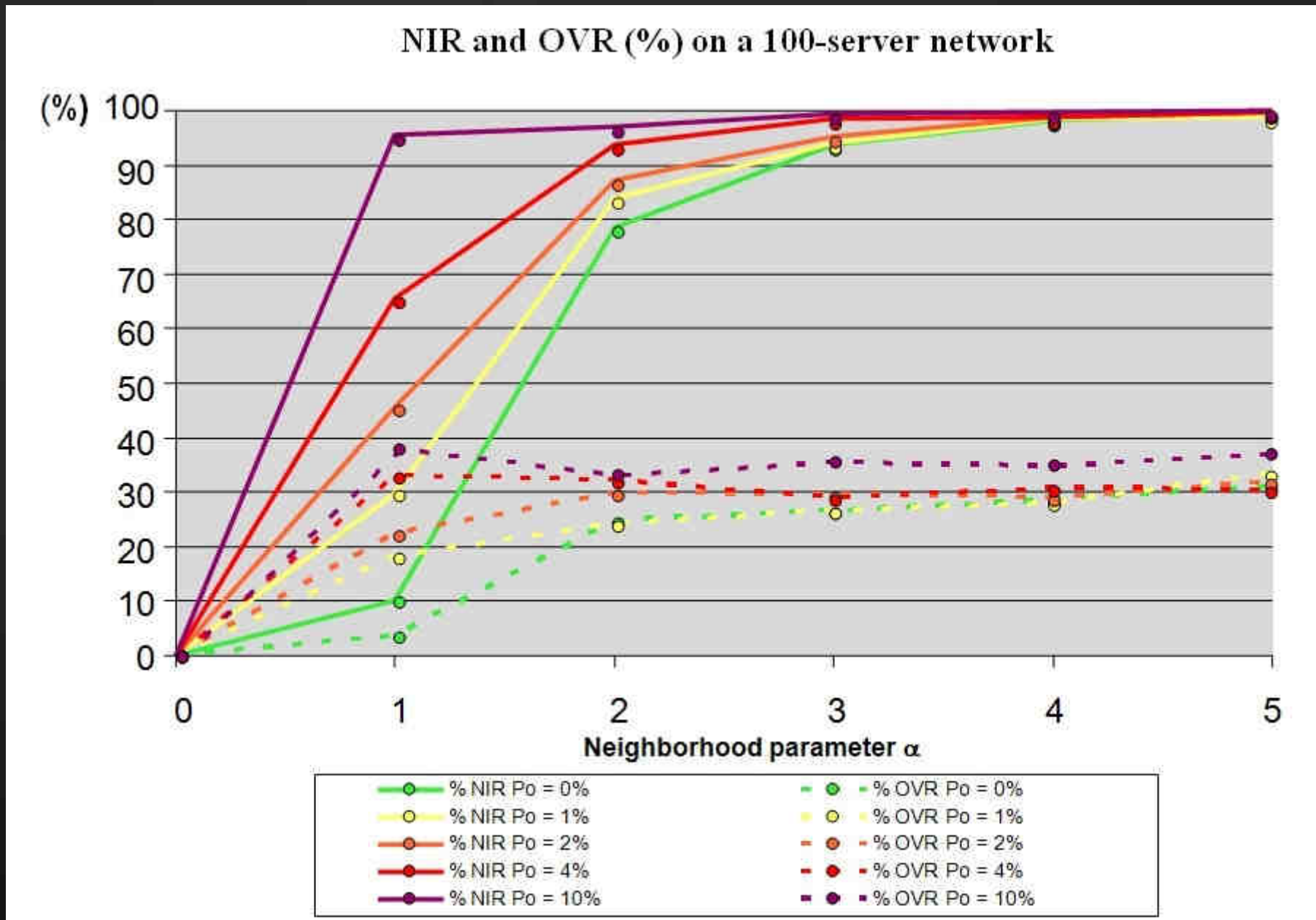
- Le sous-ensemble de nœuds {2, 4, 5} est vertex cover de G. De plus, c'est le plus petit possible.

Administration du réseau infecté (5)

- A partir des données collectées, l'attaquant cherche une structure de vertex cover $V' = \{n_{i1}, \ldots, n_{ik}\}$. Il peut considérer un sous-graphe partiel.
  1. Les informations d'administration (pour gérer ou piloter le botnet) sont envoyées seulement aux nœuds $n_{ij} \in V'$ with $1 \leq k$, only.
  2. Chacun des noeuds $n_{ij} \in V'$ va alors localement propager l'information aux autres nœud du graphe selon un ordre optimisé (par exemple, dans le graphe précédent, le nœud 3 qui peut être administré soit par le nœud 2 soit par le le noeud 4, sera administré par le nœud 2 uniquement).
- L'utilisation d'une structure de vertex cover minimise le nombre de communications entre les nœuds tout en couvrant tous les nœuds simultanément.
- Du côté du défenseur, le problème est beaucoup plus complexe à résoudre comme il ne dispose pas en totalité des données collectées par l'attaquant.

Simulation et résultats

- Conception du simulateur Suwast (*Super Worm Analysis and Simulation Tool*).
- Outil de simulation de réseaux hétérogènes complexes (clients, serveurs, routeurs...) permettant les simulations d'attaques réseaux, au niveau paquets, dans un environnement contrôlé.
- Simulations à large échelle (réseau jusqu'à 60 000 hôtes sur une machine 2 Gb).
- Possibilité d'interconnecter plusieurs machines pour simuler des millions d'hôtes (« Internet dans une éprouvette »).


NIR and OVR (%) on a 100-server network
(%)
100
90
80
70
60
50
40
30
20
10
0
0
1
2
3
4
5
Neighborhood parameter α
% NIR Po = 0%
% NIR Po = 1%
% NIR Po = 2%
% NIR Po = 4%
% NIR Po = 10%
% OVR Po = 0%
% OVR Po = 1%
% OVR Po = 2%
% OVR Po = 4%
% OVR Po = 10%

Simulation et résultats (3)

- Trois résultats principaux :
  - le paramètre p_0 a un impact significatif à la fois sur NIR et OR. Le cas $p_0 = 0.04$ est optimal, tant que le paramètre de voisinage α de serveurs est relativement limité ;
  - NIR est systématiquement supérieur à 90 % si $3 \leq \alpha$ (paramètre de voisinage réseau), la plupart des résultats étant proches de 99 % ;
  - Le paramètre de voisinage réseau α a un impact significatif plus important sur OR. Optimalement nous avons

$$\alpha \in [3, 6].$$

Conclusion

- Il existe une infinité de manières de concevoir des malwares (voir bibliographie).
- Quel est le niveau de risque actuellement ?
  - Difficile ou impossible à dire !
  - Potentiellement élevé pour les attaques ciblées (agences de renseignement, forces armées de certains pays…).
  - Probablement faible à moyenne pour les autres acteurs… jusqu'à présent !
  - Requiert des concepteurs de malware avec un bon niveau en mathématiques, en informatique et en programmation).

Conclusion

- La solution pour lutter contre ces nouvelles menaces futures n’est plus technique et ne le sera jamais !
- Seules des modèles et des politiques de sécurité sont susceptibles de constituer la meilleure
  - Evitez d’être infecté ou vous êtes mort !

Merci de votre attention

Questions ?

Bibliographie

- E. Filiol (2007) *Techniques virales avancées*, collection IRIS, Springer Verlag. An English translation is pending.
- P. Beaucamps & E. Filiol. On the possibility of practically obfuscating programs - Towards a unified perspective of code protection. WTCV'06 Special Issue, G. Bonfante & J.-Y. Marion eds, *Journal in Computer Virology*, 3 (1), 2007.
- E. Filiol & S. Josse. A Statistical Model for Viral Detection Undecidability. EICAR 2007 Special Issue, V. Broucek ed., *Journal in Computer Virology*, 3 (2).
- E. Filiol. Formalization and Implementation Aspects of K-ary (malicious) Codes. EICAR 2007 Special Issue, V. Broucek ed., Journal in Computer Virology, 3 (2).
- E. Filiol. Metamorphism, Formal Grammars and Undecidable Code Mutation. *International Journal in Computer Science*, 2 (1), pp. 70--75, 2007.
- E. Filiol, E. Franc, A. Gubbioli, B. Moquet & G. Roblot. Combinatorial Optimisation of Worm Propagation on an Unknown Network. *International Journal in Computer Science*, 2 (2), pp. 124--130, 2007.
- E. Filiol & F. Raynal. Malicioux Cryptography ... reloaded and also malicious statistics. CanSecWest 2008, Vancouver, March 26th - 28th, 2008.