The control of Technology in the Light of Snowden's Leaks

CRYPTO HOT CASES ONE YEAR BACKWARD – PART II

Eric Filiol efiliol@netc.fr

Operational cryptology and Virology Lab (C + V)°

- Due to a priority and critical mission for the French Dep.t of Justice, I regret not to be able to give my keynote speak. I beg you to excuse me and I would like to thank Baptiste DAVID for having accepted to present my talk.
- Have a nice C0c0n event!

Agenda

- Introduction: The Past and Present Context
  - 70 years of Crypto control
  - Is Snowden really the first whistleblower?
  - The context
- Case I – Connected People
  - Issues like RSA Dual_EC_DRBG, Bullrun, Heartblead, Google vs ANSSI…
- Windows Oddities
  - Beyond vulnerabilities
- Case II – Non Connected People/Non-classical Environments
  - Issues like TAO project and hardware bugs, sophisticated malware…
- Conclusion
  - What will be the future?
  - How could we resist?

Introduction

70 years of Crypto Control – Snowden and Predecessors – The Context.

Introduction

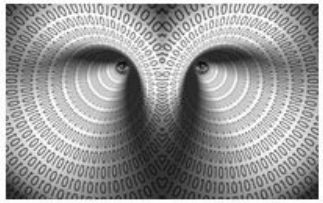
- A few weeks before Snowden's first leaks, I wrote a long study about the control of technology including cryptology (presented later at HIP 2013) [1].
- In my past experience, I have exploited partly this control (mathematical backdoors) over cryptology for cryptanalysis purposes (in the context of Hans Buehler's case).
- This control is enforced since late 1940's! So Snowden's leaks are not really new !
  - Vladimir Vetrov (aka *Farewell*) (1980)
  - Peter Wright's "Spycatcher" (1985). Former MI5's deputy director
  - Cryptome website
  - Many others…

ible for the Massacres In Ukraine? Justifying the Unjustifiable. How the UN Defines "Human Rights"

Swedish Intelligence Service Spying on Russia for US National Security Agency

By Jordan Shilton
Global Research, December 30, 2013
World Socialist Web Site

Region: Europe, Russia and FSU
Theme: Intelligence



Documents released by Edward Snowden reveal that Sweden's National Defence Radio Establishment (FRA) has been collecting large quantities of communications data from Russia, which it has passed to the American National Security Agency (NSA).

The revelations confirm that such collaboration goes back for decades.

According to one document published by public broadcaster SVT December, Stockholm signed a top-secret cooperation agreement in 1954 with the United States, Britain, Australia, New Zealand and Canada—the so-called Five Eyes—to exchange intelligence information. This was replaced in 2004 by bilateral agreements with each country, which saw the collaboration intensify. Throughout the Cold War, FRA passed information obtained from the Soviet Union to its Western allies.

"The relationship with Sweden is protected on the top-secret level because of the country's political neutrality," a 2006 NSA document noted.

Due to its geographic location, the FRA has access to Russian communications, including those from "high priority" targets from politics and areas of economic interest, such as the energy sector. Estimates from *Russia Today* put the amount of communications data from Russia that passes through Sweden at 80 percent. A recent NSA document from April 2013 states, "FRA provided NSA (with a) unique collection on high-priority Russian targets, such as leadership, internal politics."

The cooperation between FRA and the NSA was expanded significantly in 2011, with the NSA gaining access to FRA's network of cables. This included the ability to intercept communications from the Baltic countries through under-sea cables controlled by Sweden.

Font-size: A Print:

Latest News / Top Stories

- Is the West "Directly" Responsible for the Massacres In Ukraine?
- Oligarch Ihor Kolomoyskyi: Washington's "Man in Ukraine"
- Justifying the Unjustifiable. How the UN Defines "Human Rights"
- America Brings Hell to Ukraine as Part of its Plan for World Domination
- Turkish Mine Explosion, Predicted in 2010
- Fatal Car Crashes in America: US Government Whitewashes GM's Responsibility for Deadly Ignition Defect
- Ukraine: Oligarch Ihor Kolomoisky Offers $1 Million to Murder Federalist Leader Oleg Tsarev

All Articles

NEWS
MOST POPULAR
GEOGRAPHIC REGIONS
THEMES
I-BOOKS SERIES
IN-DEPTH REPORTS
THE GLOBAL RESEARCH NEWS HOUR

The Context

- The control techniques depend on the target context/environment

Type	Data	NSA Programs	Techniques	Examples
Connected	Plaintext	PRISM, Xkeyscore…	Data collection, wiretapping, eavesdropping, agreements with industry/providers….	Google, Facebook, Apple, Microsoft (including Skype)…
	Ciphertext	Bullrun/Edgehill…	Malware, 0-day exploitation, random generator control, security standards control, controlling CAs, bugging software, applied cryptanalysis…	Heartbleed, RSA, Google/ANSSI, Mail.ru, Alibaba…
Connected by private network	Ciphertext	Cottonmouth, Godsurge, TOR attack, Quantum, Foxacid, Firework, Bulldozer…	Malware, 0-day exploitation, random generator control, controlling CAs, security standards control, bugging software, hardware bugs, mathematical trapdoors…	TOR network, Gasprom, Petrobras, French MFA, Aeroflot, Total. Airbus, SWIFT…
Non-connected (offline)	Ciphertext	TAO, still unknown projects???	Tempest techniques, mathematical backdoors, hardware bugging, Humint	Hans Buehler Case (1995). Gov, MIL, Sensitive companies

CASE I – Spying Connected people

Issues like Bullrun, RSA Dual_EC_DRBG, Heartblead, Windows oddities…

Bullru

- Goal: bypas
- Applied cry
  - Tampering promote Dual_EC_D
  - Influencing bar strong
  - Working wi random nu
  - Attacking t
  - Identifying
  - Establishin telecommu
  - Bypassing
- Annual bud

d) to (e.g

ng to

and

lobal

theguardian

News | Sport | Comment | Culture | Business | Money | Life & style | Travel | Environment | T

News › World news › The NSA files

Series: Glenn Greenwald on security and liberty

Previous | Next | Index

Revealed: how US and UK spy agencies defeat internet privacy and security

- NSA and GCHQ unlock encryption used to protect emails, banking and medical records
- $250m-a-year US program works covertly with tech companies to insert weaknesses into products
- Security experts say programs 'undermine the fabric of the internet'

- Q&A: submit your questions for our privacy experts

James Ball, Julian Borger and Glenn Greenwald
Guardian Weekly, Friday 6 September 2013

Jump to comments (4142)

Article history

The NSA Files: Decoded

Through covert partnerships with tech companies, the spy agencies have inserted secret vulnerabilities into encryption software. Photograph: Kacper Pempel/Reuters

What the surveillance revelations mean for you

World news
The NSA files · NSA · Surveillance · United States · US national security · Privacy

UK news
GCHQ

Technology

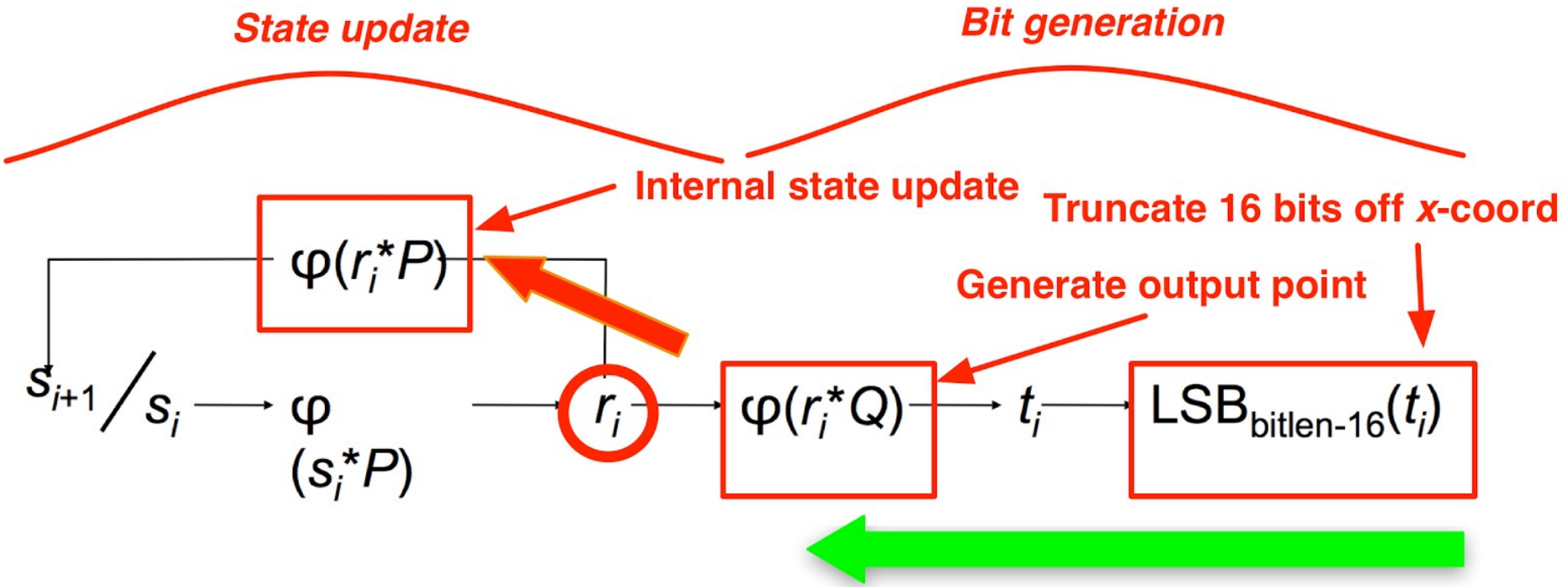
US and British intelligence agencies have successfully cracked much of

Dua

A.1 Constants for the Dual_EC_DRBG

The **Dual_EC_DRBG** requires the specifications of an elliptic curve and two points on the elliptic curve. One of the following NIST **approved** curves with associated points **shall** be used in applications requiring certification under [FIPS 140]. More details about these curves may be found in [FIPS 186]. If alternative points are desired, they **shall** be generated as specified in Appendix A.2.

Each of following curves is given by the equation:



fro

- In
ha

```
      f4a13945 d898c296

Py = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece
     cbb64068 37bf51f5

Qx = c97445f4 5cdef9f0 d3e05e1e 585fc297 235b82b5 be8ff3ef
     ca67c598 52018192

Qy = b28ef557 ba31dfcb dd21ac46 e2a91e3c 304f44cb 87058ada
     2cb81515 1e610046
```

Dual_EC_RDBG Timeline

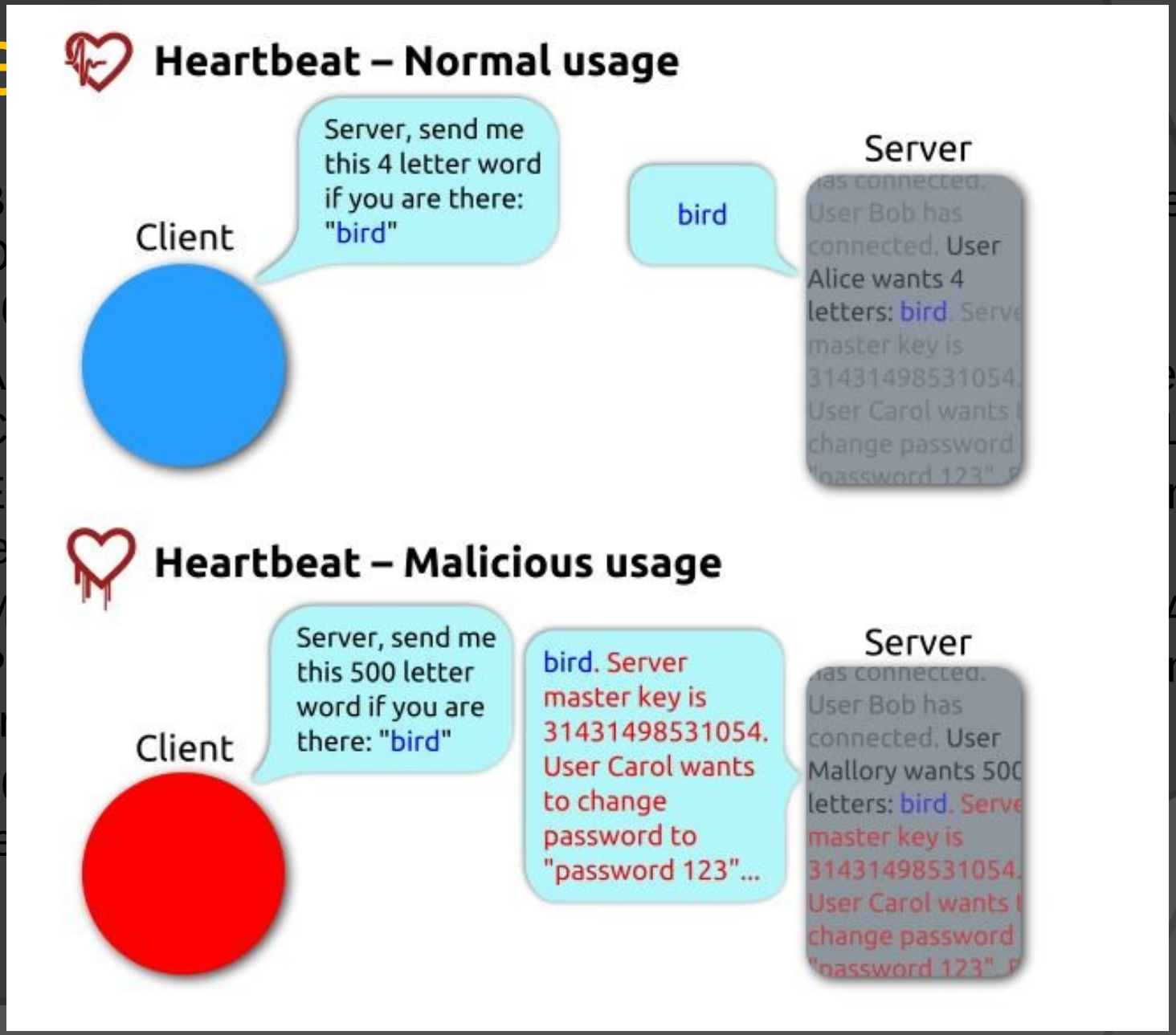
- 2004 - RSA makes Dual_EC_DRBG the default CSPRNG in BSAFE
- 2005 - ISO/IEC 18031:2005 is published, and includes Dual_EC_DRBG. The first draft of NIST SP 800-90A is released to the public, includes Dual_EC_DRBG
- 2006 – 2007 – Works suggesting the existence of a NSA backdoor (K. Gjosteen, Berry Schoenmakers and Andrey Sidorenko, Shumow/Fergusson…)
- June 2006 - NIST SP 800-90A is published, includes Dual_EC_DRBG with the defects pointed out by Kristian Gjøsteen and Berry Schoenmakers and Andrey Sidorenko not having been fixed.
- June/Sep. 2013 – Snowden leak about Bullrun and Dual_EC_DRBG
- 19 Sep. 2013 - RSA Security advises its customers to stop using Dual_EC_DRBG in RSA Security's *BSAFE* toolkit
- Dec. 2013 - Reuters reports this is a result of a secret $10 million deal with NSA
- April 21st, 2014, Following a public comment period and review, NIST removed Dual_EC_DRBG as a cryptographic algorithm from its draft guidance on random number generators, recommending "that current users of Dual_EC_DRBG transition to one of the three remaining approved algorithms as quickly as possible

Hot Issue

- Specific subtle formulation in the NIST standard meant that you could only get the crucial FIPS 140-2 validation (Cryptographic Module Validation Program) of your implementation if you used the original compromised *P* and *Q* values
- This includes the FIPS 140-2 statistical test suite (now NIST STS) which are THE *de facto* world standard for cryptography statistical evaluation/validation
  - Passing successfully the tests does mean your generator is secure
- Up to me, FIPS 140-2 tests are “backdoored” (they are purposely non significant enough by not including a few additional testing techniques)
- Issue of statistical test simulability (Filiol, 2006): “*if I know your tests, I can simulate and bypass them*”
- Cryptography statistical validation should use a secret national process/set of tests (as it is the case in France)

He

- B … e in O … 14th, 2…
- A … e and C … 14
- E … mory le…
- M … Afee, P … niper fi…
- 3… few re…



NSA gets early access to zero-day data from Microsoft, others

Meant to help secure network, data could be used to attack foreign governments

Microsoft Bugs

In addition to private communications, information about equipment specifications and... **Read More**

Microsoft Corp. (MSFT), the world's largest software company, provides intelligence agencies with information about bugs in its popular software before it publicly releases a fix, according to two people familiar with the process. That information can be used to protect government computers and to access the computers of terrorists or military foes.

Redmond, Washington-based Microsoft and other software or Internet security companies have been aware that this type of early alert allowed the U.S. to exploit vulnerabilities in software sold to foreign governments, according to two U.S. officials. Microsoft doesn't ask and can't be told how the government uses such tip-offs, said the officials, who asked not to be identified because the matter is confidential.

Frank Shaw, a spokesman for Microsoft, said those releases occur in cooperation with multiple agencies and are designed to give government "an early start" on risk assessment and mitigation.

In an e-mailed statement, Shaw said there are "several programs" through which such information is passed to the government, and named two which are public, run by Microsoft and for defensive purposes.

Photographer: Scott Eells/Bloomberg

Microsoft Corp., the world's largest software

Microsoft's single largest customer, systems on both its unclassified and secret networks (NIPRNET and SIPRNET) use Microsoft software. Microsoft has similar early-access programs for other customers, and it often deploys patches to large customers for testing prior to pushing them out on its monthly "Patch Tuesday" schedule.

Windows Oddities

Beyond vulnerabilities…

Are Vulnerabilities Really Necessary?

- The design of systems can enable the use of dynamic resources that can
  - transparently,
  - without any evidence/traces let into the system,
  - for a limited period of time

  be added to the system with preemptive rights
- Let us have a short journey into Windows for having an idea of what could be done
- Used in Cryptographic Dynamic Backdoors (my talk at CanSecWest 2011) among many other possibilities
- Everything occurs in memory only using legitimate Windows mechanisms

Shim Mechanism

- What a shim is?
  - A shim is used in a context of backward compatibility
  - Windows operating system evolves from version to version: new technology added (ASLR, DEP…), bug fixes, modification in strategy…
  - It could lead to compatibility issues for a few software
    - In case of ASLR, if a software imported one function from Windows API by a fixed address (e.g. *LoadLibraryA* located at *0x7c801d77* in *Kernel32.dll* in Windows XP)
    - Some features of software were based on bugs ("*it works by mistake*" software)… If these ones are fixed, these applications will not remain the same
  - Shims simulate the behavior of older versions of Windows for legacy applications that rely on incorrect or deprecated functionality, or correct the way in which poorly written applications call unchanged APIs.

Shim Mechanism

From a technical point of view:

- One solution to keep backward compatibility is placing branches directly in the source code for Windows but it will not be a good solution (challenge for the serviceability and reliability of Windows source code)

  -> Shims were born !
- Technically, shim infrastructure implements a form of API hooking. It redirects API calls from Windows itself to alternative code—the shim itself.
- This redirection is automatically performed from IAT section in the MZ-PE file format

Shim Mechanism

Why using shim?

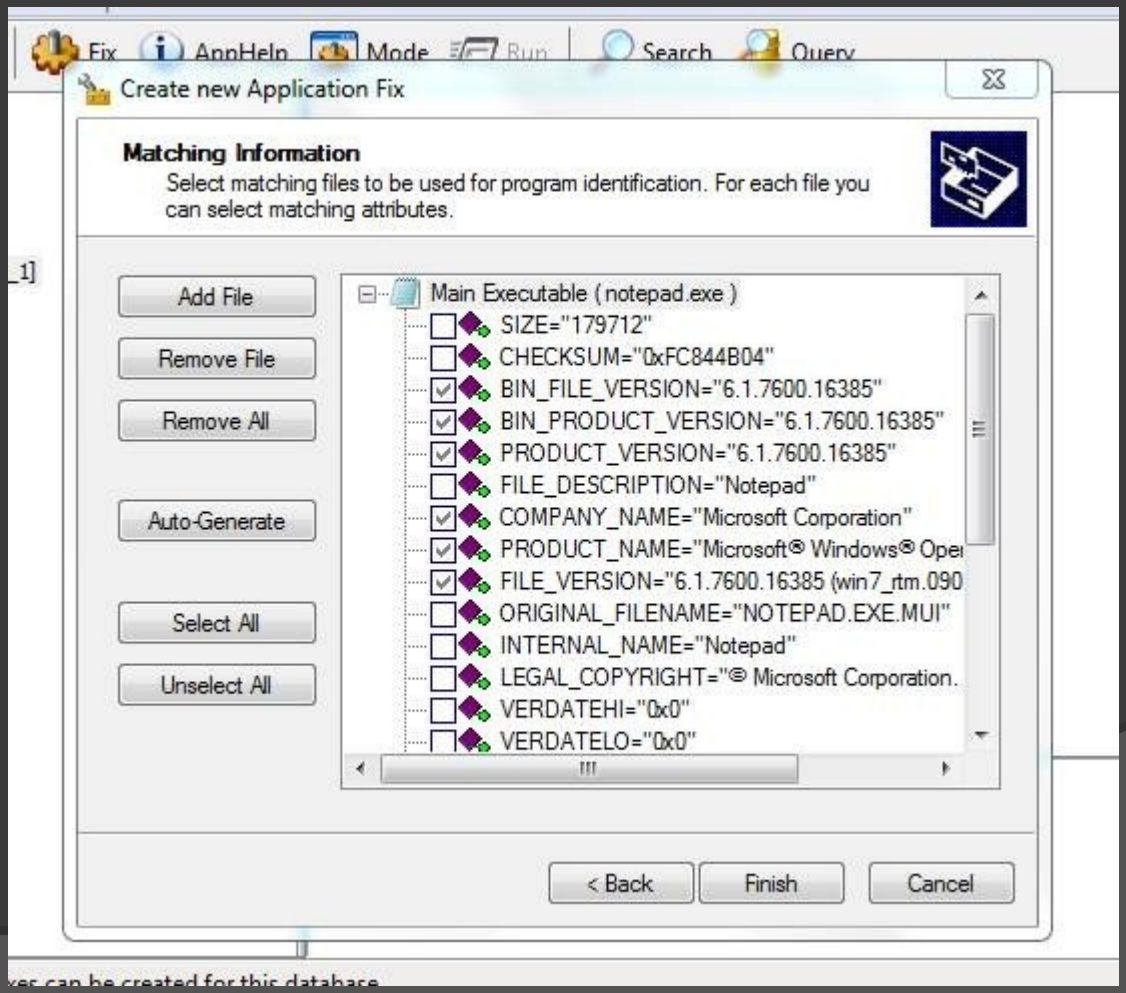
- You can fix applications without access to the source code, or without changing them at all
- Most people typically consider shims for applications where the vendor is out of business nor they don’t want to spent to much money in an update
- Use it temporally will provide to users a shimmed and functional version which can bridge a gap until a compatible version is available
- To make a transition between two major versions of Windows (XP to Seven)

Shim is not always a solution:

- The downside is that most vendors do not support shimmed applications. You cannot fix every application using shims.

Shim Mechanism

How linking a shim to an application [6]?



Shim Mechanism

What Microsoft tells you about:

- It does not change security since:
  - Application redirected to the shim prior to calling Windows, the code that runs inside a shim still sits outside Windows. Consequently, Windows holds shim code to the same security restrictions as the application code itself
  - For example, no shim is available to bypass the Windows 7 UAC prompt while still running the application with elevated permissions. (…) The same is true for code that you write yourself
  - Therefore, when evaluating the security implications of using shims in your enterprise, you are not opening any additional security vulnerability.
  - Shims run as user-mode code inside a user-mode application process, you cannot use a shim to fix kernel-mode code
- Shim Infrastructure, in essence, injects additional code into the application before it calls into Windows API, (…) [*it*] could be done by modifying the application code itself in fact

Shim Mechanism

What must be understood (**what Microsoft does not tell you**):

- Suppose an application ran using (willingly or not) a vulnerability in the Windows API (buffer overflow, execution from data or stack, ASLR…), if this one is fixed but the shim still allows the application to work with … what must be concluded?
  - Exploits on a specific software can be “maintained”…
  - It fixes vulnerabilities in Windows but it could add some in software
- This detour of API can be redirected to deprecated or undocumented API with less control checks than the regular one, even if Microsoft says no.
- For encryption software, you can use redirection of specific API for bad purposes. If Windows encryption is used, it is possible at that point to add a bias in keys used (or send them to you). In fact, shim is just *another legal hook*
- Even if this mechanism is not used for drivers, drivers are still MZ-PE executables and can/could be targeted by similar mechanisms in the future.

Perverting Windows Cryptography

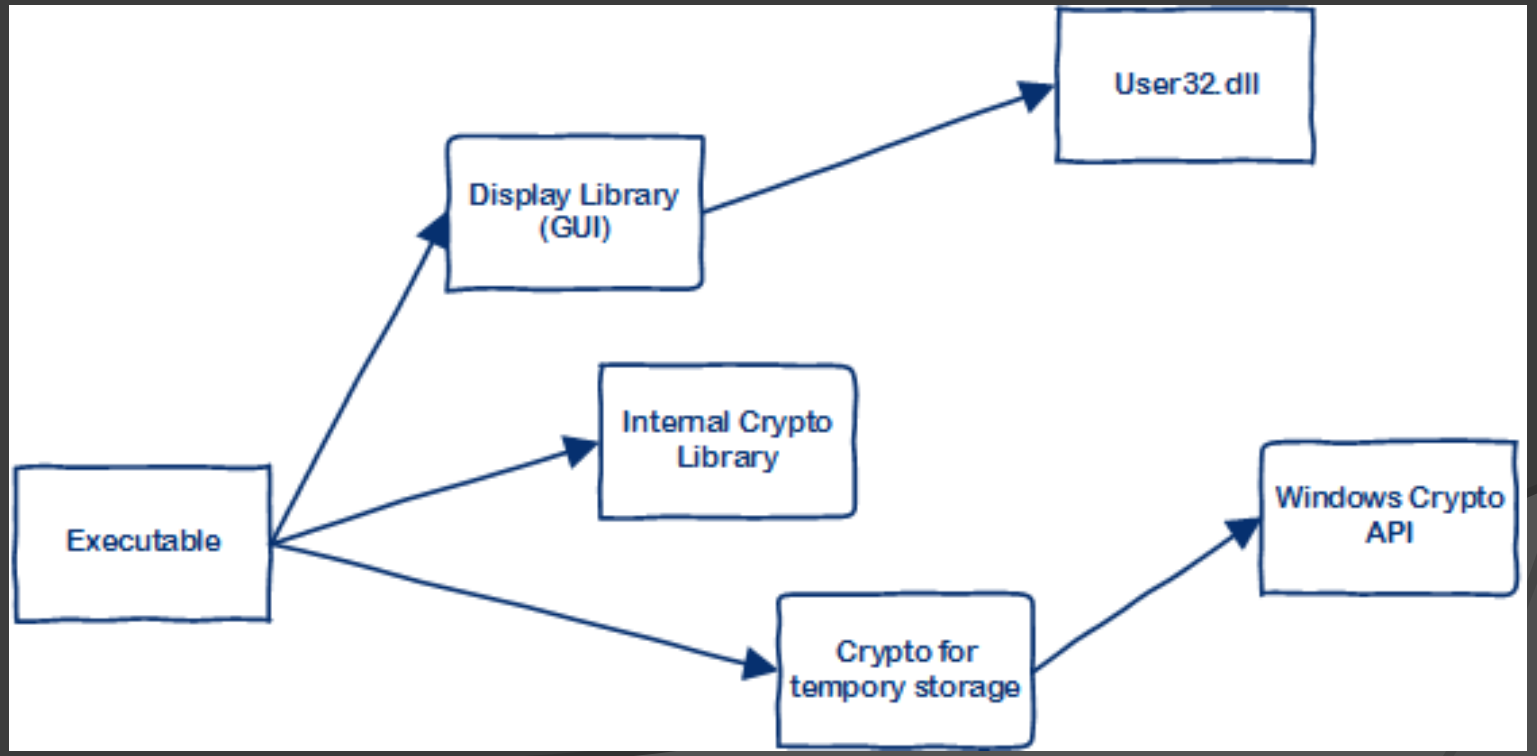
Context: *Dynamic Cryptographic Backdoors* (CanSecWest 2011)

- The goal is to:
  - Trap cipher features in a software.
  - Change normal behavior of random generators
  - Force “outsourcing” of sensitive data
  - Ask the “good” questions…
    - Once we have got access to cipher routines, we can read plaintext, modify cryptographic keys, modify cipher algorithm used on-the-fly (use a temporally weakest one), make sensitive data eavesdrop (plaintext, secret keys…)
- Applied to and tested successfully with the AES in different environments

Perverting Windows Cryptography

Attack of a software which externalizes its cipher routines

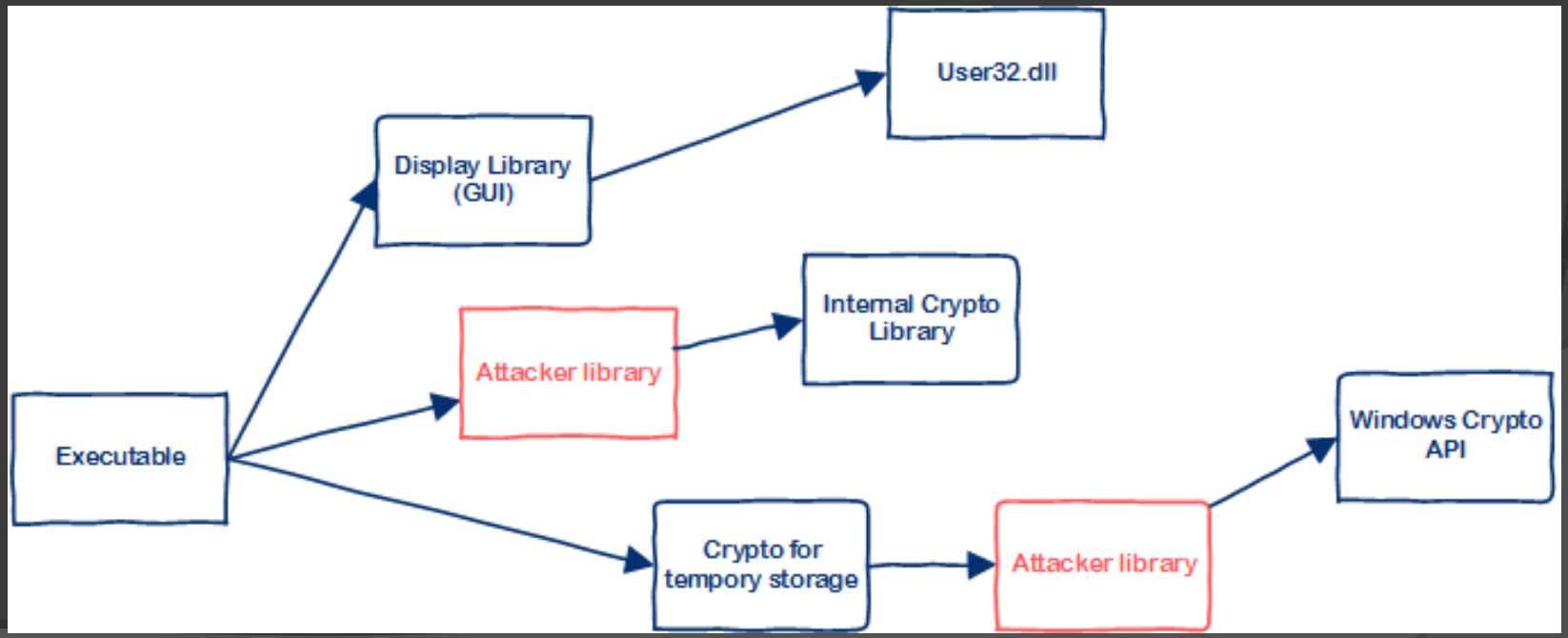
- Either it can use its own library of cipher routines (.dll)
- Or it can use Windows crypto API.



Perverting Windows Cryptography

Attack of a software which externalize its cipher routines. Several methods to attack:

- Hook/detour on the internal library of the software containing the cipher features of the software
- Hook/detour the API of Windows used by the software (think about shims which can be used in such a case)



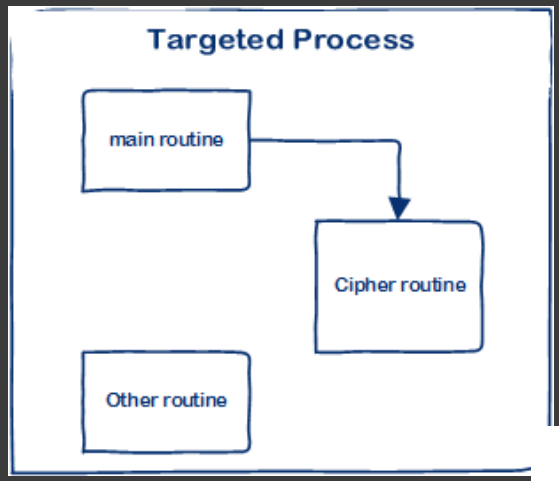
Perverting Windows Cryptography

Attack of a software which uses its internal cipher routines

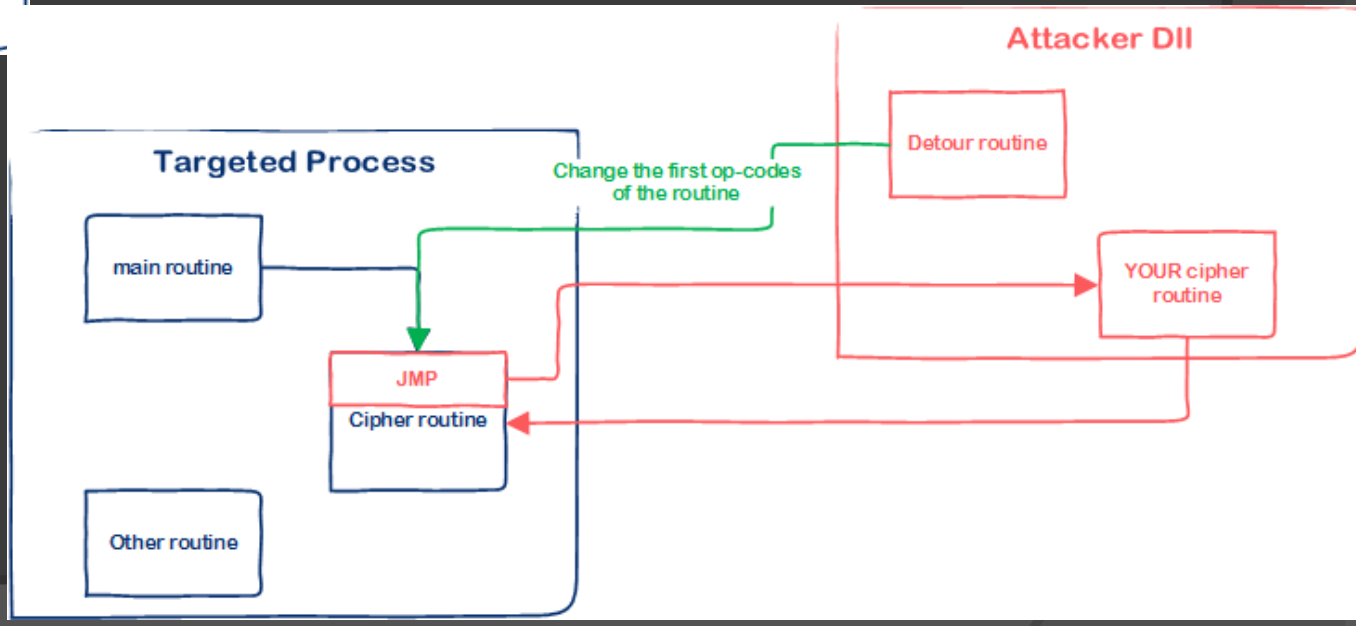
- Since the cipher procedure is stored inside the software, it *should not* be possible to intercept every call as we did previously
- But it is still possible to
  1. reverse in order to locate cipher routines in the software
  2. Create a Dll which is able to patch first op-codes of these routines to perform a jump operation inside routines in your Dll. This Dll will be injected in the targeted software and is supposed to “detour” the internal function to that inside it
  3. Do whatever you want in your Dll and return the flow of execution to the process in the original routine
- You can use a shim as a legitimate dll store/provider (everything occurs in memory only)

Perversion on Windows Cryptography

Normal flow of execution



Hijacked execution flow



Perverting Windows Cryptography

Inside Windows crypto API

- *CryptProtectMemory* function: protect some sensitive data in memory (e.g passwords) but strangely no password is required to manage this protection!
- *CryptUnprotectMemory* function is used to retrieve information. Same prototype used to call the function
- There is no key/entropy source field! False sense of security!

CryptProtectMemory function

The **CryptProtectMemory** function *encrypts* memory to prevent others from viewing sensitive information in your process. For example, use the **CryptProtectMemory** function to encrypt memory that contains a password. Encrypting the password prevents others from viewing it when the process is paged out to the swap file. Otherwise, the password is in *plaintext* and viewable by others.

Syntax

C++

```
BOOL WINAPI CryptProtectMemory(
  _Inout_  LPVOID pData,
  _In_     DWORD cbData,
  _In_     DWORD dwFlags
);
```

Perverting Windows Cryptography

Where the password/keys is ???

- In fact, it is a source of entropy generated from kernel with some information available only in kernel mode (cookies, salt, creation time...)
- Different possibility to use it:
  - CRYPTPROTECTMEMORY_SAME_PROCESS
  - CRYPTPROTECTMEMORY_CROSS_PROCESS
  - CRYPTPROTECTMEMORY_SAME_LOGON
- The best protection is the one which deals with one process. In such a case, information is only valid fort the current process which calls this API. Any read access from other processes in its memory will see encrypted data
- But, if you succeed at injecting code inside the process, you can read it normally
- From kernel land, it is a piece of cake to read *protected data*.

Perverting Windows Cryptography

Funny story about CryptProtectMemory

- Story from http://blog.gentilkiwi.com/tag/cryptprotectmemory
- This API is very old (it has been introduced with XP/2003)
- *Lsass.exe* stores all the password for windows when you log in
- It seems that Microsoft tries hard to improve sensitive and historic routines.... but does not always use it!

```
dword_8E0880 dd 70h                     ; DATA XREF: debug132:01206438↓o
dd 16h
_UNICODE_STRING <20h, 22h, offset aSekurlsa@liv_0> ; "sekurlsa@live.fr"
_UNICODE_STRING <16h, 18h, offset aPsPassword_0> ; "ps:password"
_UNICODE_STRING <14h, 16h, offset a?OczJqi_uFwp0> ; "?%*ÄccÌ¦âqê\a(.ôÀ¤+\nfwP0]"
```

Under Windows 8 : Passwords are encrypted

```
dword_B2CE30 dd 96h                     ; DATA XREF: debug045:00F7BAC0↓o
dd 14h
_UNICODE_STRING <20h, 22h, offset aSekurlsa@liv_0> ; "sekurlsa@live.fr"
_UNICODE_STRING <16h, 18h, offset aPsPassword_0> ; "ps:password"
_UNICODE_STRING <14h, 16h, offset aWaza1234> ; "waza1234//"
```

Under Windows 8.1 : Passwords are not!

Random Generation API in Ring 0

In many cases, it can be useful to access to random generator

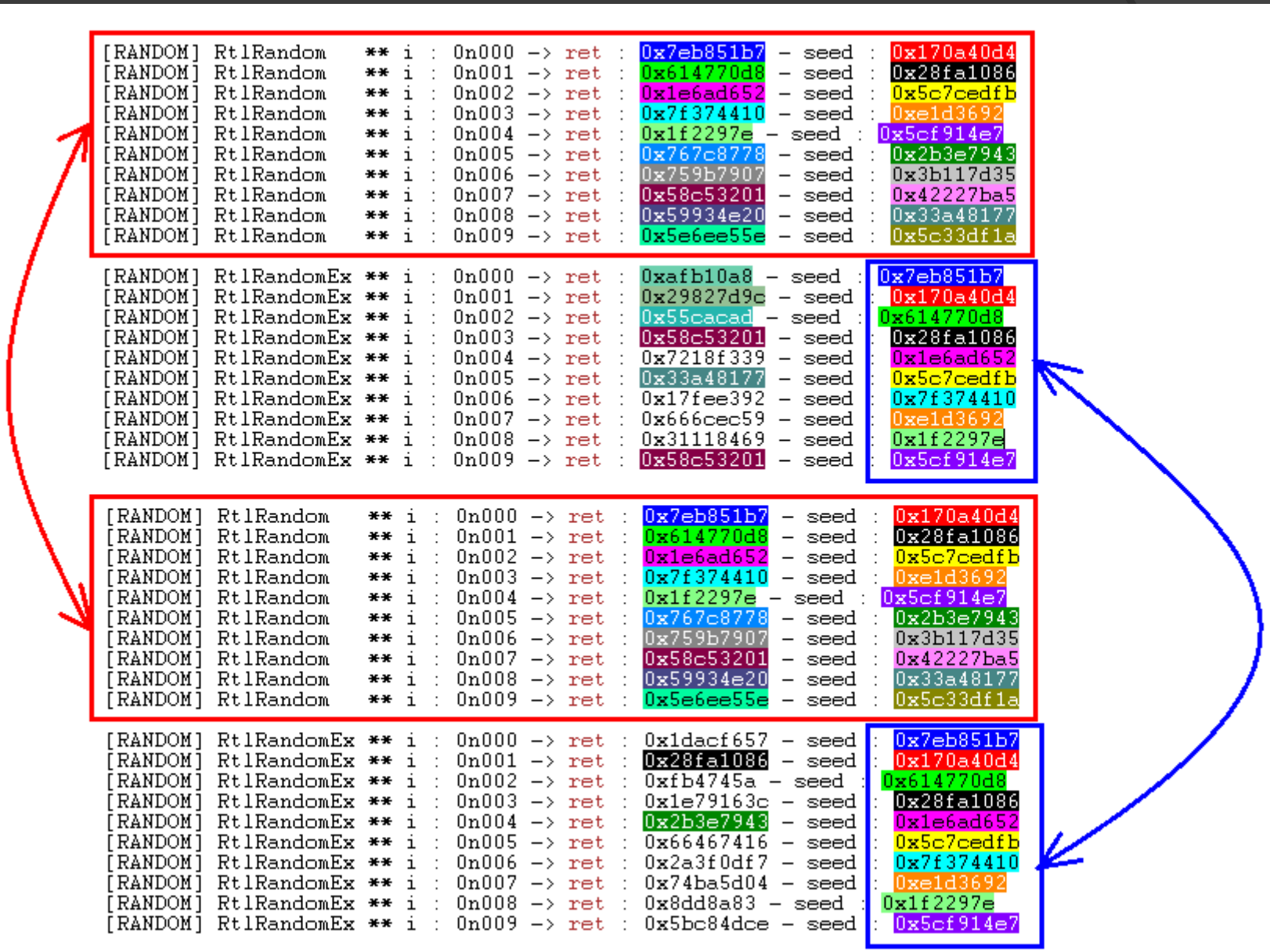
- In ring 0, at least two routines are available (are there more?): *RtlRandom* and *RtlRandomEx*.
- From the Msdn documentation :
  - *RtlRandom* returns values that are uniformly distributed over the range from zero to the maximum possible LONG value minus 1 if it is called repeatedly with the same *Seed*.
  - The *RtlRandomEx* function is an improved version of the *RtlRandom* function that is twice as fast and produces “better” random numbers.
- These routine could be used for :
  - Random event trigger : security software want to test something “sometime”…
  - A very bad random source for cryptography however…

Random API in Ring 0

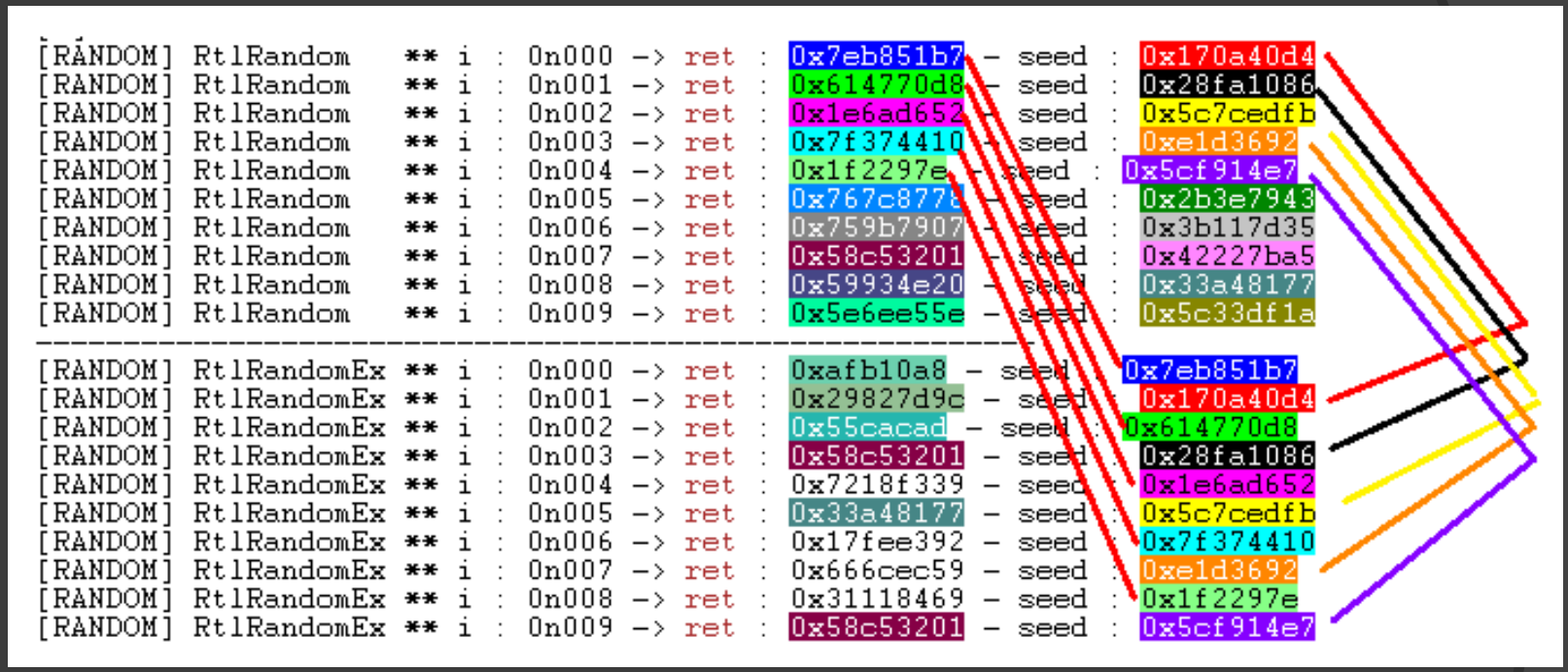
Let us do “black box” tests with these generators to evaluate their real randomness quality

- Let’s test generators with a fixed seed
- We test on both generators evolution of seed and return value
- Tests are launched twice with several seconds between them

```
VOID SupGenerateRandomNumber(void){

    ULONG i, seed, ret;

    seed = 0x123456;
    ret = 0;

    for(i = 0; i < 10; i++){
        ret = RtlRandom(&seed);
        DebugPrint(TRACE_LEVEL_INFORMATION, FUNCTION, "[RANDOM] RtlRandom   ** i : 0n%03d -> ret : %#I64x - seed : %#I64x", i, ret, seed);
    }

    DebugPrint(TRACE_LEVEL_INFORMATION, FUNCTION, "-------------------------------------------------- -----");

    seed = 0x123456;
    ret = 0;

    for(i = 0; i < 10; i++){
        ret = RtlRandomEx(&seed);
        DebugPrint(TRACE_LEVEL_INFORMATION, FUNCTION, "[RANDOM] RtlRandomEx ** i : 0n%03d -> ret : %#I64x - seed : %#I64x", i, ret, seed);
    }

}
```

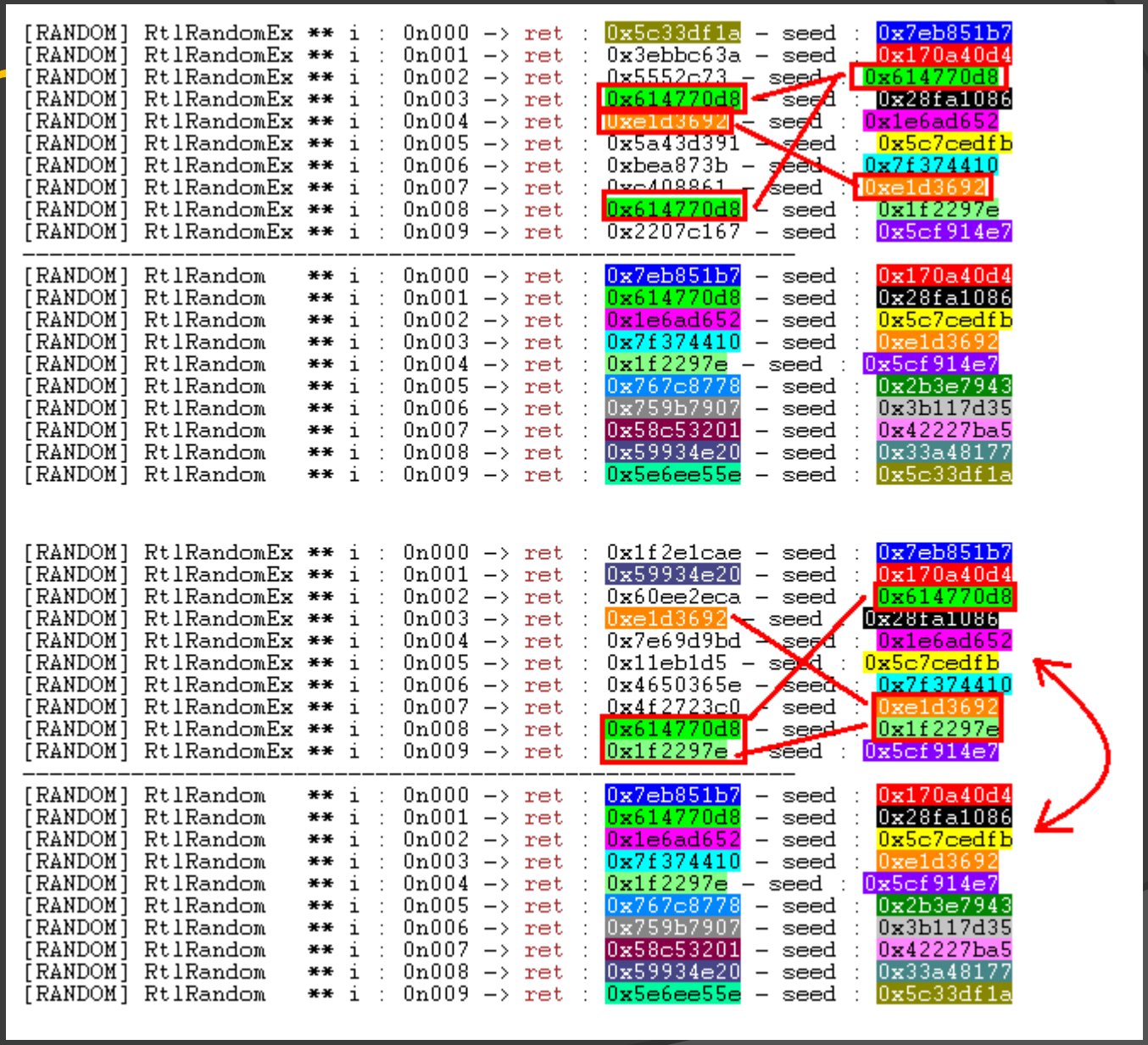

[RANDOM] RtlRandom ** i : 0n000 -> ret : 0x7eb851b7 - seed : 0x170a40d4
[RANDOM] RtlRandom ** i : 0n001 -> ret : 0x614770d8 - seed : 0x28fa1086
[RANDOM] RtlRandom ** i : 0n002 -> ret : 0x1e6ad652 - seed : 0x5c7cedfb
[RANDOM] RtlRandom ** i : 0n003 -> ret : 0x7f374410 - seed : 0xe1d3692
[RANDOM] RtlRandom ** i : 0n004 -> ret : 0x1f2297e - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n005 -> ret : 0x767c8778 - seed : 0x2b3e7943
[RANDOM] RtlRandom ** i : 0n006 -> ret : 0x759b7907 - seed : 0x3b117d35
[RANDOM] RtlRandom ** i : 0n007 -> ret : 0x58c53201 - seed : 0x42227ba5
[RANDOM] RtlRandom ** i : 0n008 -> ret : 0x59934e20 - seed : 0x33a48177
[RANDOM] RtlRandom ** i : 0n009 -> ret : 0x5e6ee55e - seed : 0x5c33df1a
[RANDOM] RtlRandomEx ** i : 0n000 -> ret : 0xafb10a8 - seed : 0x7eb851b7
[RANDOM] RtlRandomEx ** i : 0n001 -> ret : 0x29827d9c - seed : 0x170a40d4
[RANDOM] RtlRandomEx ** i : 0n002 -> ret : 0x55cacad - seed : 0x614770d8
[RANDOM] RtlRandomEx ** i : 0n003 -> ret : 0x58c53201 - seed : 0x28fa1086
[RANDOM] RtlRandomEx ** i : 0n004 -> ret : 0x7218f339 - seed : 0x1e6ad652
[RANDOM] RtlRandomEx ** i : 0n005 -> ret : 0x33a48177 - seed : 0x5c7cedfb
[RANDOM] RtlRandomEx ** i : 0n006 -> ret : 0x17fee392 - seed : 0x7f374410
[RANDOM] RtlRandomEx ** i : 0n007 -> ret : 0x666cec59 - seed : 0xe1d3692
[RANDOM] RtlRandomEx ** i : 0n008 -> ret : 0x31118469 - seed : 0x1f2297e
[RANDOM] RtlRandomEx ** i : 0n009 -> ret : 0x58c53201 - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n000 -> ret : 0x7eb851b7 - seed : 0x170a40d4
[RANDOM] RtlRandom ** i : 0n001 -> ret : 0x614770d8 - seed : 0x28fa1086
[RANDOM] RtlRandom ** i : 0n002 -> ret : 0x1e6ad652 - seed : 0x5c7cedfb
[RANDOM] RtlRandom ** i : 0n003 -> ret : 0x7f374410 - seed : 0xe1d3692
[RANDOM] RtlRandom ** i : 0n004 -> ret : 0x1f2297e - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n005 -> ret : 0x767c8778 - seed : 0x2b3e7943
[RANDOM] RtlRandom ** i : 0n006 -> ret : 0x759b7907 - seed : 0x3b117d35
[RANDOM] RtlRandom ** i : 0n007 -> ret : 0x58c53201 - seed : 0x42227ba5
[RANDOM] RtlRandom ** i : 0n008 -> ret : 0x59934e20 - seed : 0x33a48177
[RANDOM] RtlRandom ** i : 0n009 -> ret : 0x5e6ee55e - seed : 0x5c33df1a
[RANDOM] RtlRandomEx ** i : 0n000 -> ret : 0x1dacf657 - seed : 0x7eb851b7
[RANDOM] RtlRandomEx ** i : 0n001 -> ret : 0x28fa1086 - seed : 0x170a40d4
[RANDOM] RtlRandomEx ** i : 0n002 -> ret : 0xfb4745a - seed : 0x614770d8
[RANDOM] RtlRandomEx ** i : 0n003 -> ret : 0x1e79163c - seed : 0x28fa1086
[RANDOM] RtlRandomEx ** i : 0n004 -> ret : 0x2b3e7943 - seed : 0x1e6ad652
[RANDOM] RtlRandomEx ** i : 0n005 -> ret : 0x66467416 - seed : 0x5c7cedfb
[RANDOM] RtlRandomEx ** i : 0n006 -> ret : 0x2a3f0df7 - seed : 0x7f374410
[RANDOM] RtlRandomEx ** i : 0n007 -> ret : 0x74ba5d04 - seed : 0xe1d3692
[RANDOM] RtlRandomEx ** i : 0n008 -> ret : 0x8dd8a83 - seed : 0x1f2297e
[RANDOM] RtlRandomEx ** i : 0n009 -> ret : 0x5bc84dce - seed : 0x5cf914e7

Random API in Ring 0



- The seed of RtlRandEx is an accumulation of values and seeds of RtlRandom…
- The values of RtlRandomEx are sometimes repeated
- The two generators are not so different

Rar


[RANDOM] RtlRandomEx ** i : 0n000 -> ret : 0x5c33df1a - seed : 0x7eb851b7
[RANDOM] RtlRandomEx ** i : 0n001 -> ret : 0x3ebbc63a - seed : 0x170a40d4
[RANDOM] RtlRandomEx ** i : 0n002 -> ret : 0x5552c73 - seed : 0x614770d8
[RANDOM] RtlRandomEx ** i : 0n003 -> ret : 0x614770d8 - seed : 0x28fa1086
[RANDOM] RtlRandomEx ** i : 0n004 -> ret : 0xe1d3692 - seed : 0x1e6ad652
[RANDOM] RtlRandomEx ** i : 0n005 -> ret : 0x5a43d391 - seed : 0x5c7cedfb
[RANDOM] RtlRandomEx ** i : 0n006 -> ret : 0xbea873b - seed : 0x7f374410
[RANDOM] RtlRandomEx ** i : 0n007 -> ret : 0xc408861 - seed : 0xe1d3692
[RANDOM] RtlRandomEx ** i : 0n008 -> ret : 0x614770d8 - seed : 0x1f2297e
[RANDOM] RtlRandomEx ** i : 0n009 -> ret : 0x2207c167 - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n000 -> ret : 0x7eb851b7 - seed : 0x170a40d4
[RANDOM] RtlRandom ** i : 0n001 -> ret : 0x614770d8 - seed : 0x28fa1086
[RANDOM] RtlRandom ** i : 0n002 -> ret : 0x1e6ad652 - seed : 0x5c7cedfb
[RANDOM] RtlRandom ** i : 0n003 -> ret : 0x7f374410 - seed : 0xe1d3692
[RANDOM] RtlRandom ** i : 0n004 -> ret : 0x1f2297e - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n005 -> ret : 0x767c8778 - seed : 0x2b3e7943
[RANDOM] RtlRandom ** i : 0n006 -> ret : 0x759b7907 - seed : 0x3b117d35
[RANDOM] RtlRandom ** i : 0n007 -> ret : 0x58c53201 - seed : 0x42227ba5
[RANDOM] RtlRandom ** i : 0n008 -> ret : 0x59934e20 - seed : 0x33a48177
[RANDOM] RtlRandom ** i : 0n009 -> ret : 0x5e6ee55e - seed : 0x5c33df1a
[RANDOM] RtlRandomEx ** i : 0n000 -> ret : 0x1f2e1cae - seed : 0x7eb851b7
[RANDOM] RtlRandomEx ** i : 0n001 -> ret : 0x59934e20 - seed : 0x170a40d4
[RANDOM] RtlRandomEx ** i : 0n002 -> ret : 0x60ee2eca - seed : 0x614770d8
[RANDOM] RtlRandomEx ** i : 0n003 -> ret : 0xe1d3692 - seed : 0x28fa1086
[RANDOM] RtlRandomEx ** i : 0n004 -> ret : 0x7e69d9bd - seed : 0x1e6ad652
[RANDOM] RtlRandomEx ** i : 0n005 -> ret : 0x11eb1d5 - seed : 0x5c7cedfb
[RANDOM] RtlRandomEx ** i : 0n006 -> ret : 0x4650365e - seed : 0x7f374410
[RANDOM] RtlRandomEx ** i : 0n007 -> ret : 0x4f2723c0 - seed : 0xe1d3692
[RANDOM] RtlRandomEx ** i : 0n008 -> ret : 0x614770d8 - seed : 0x1f2297e
[RANDOM] RtlRandomEx ** i : 0n009 -> ret : 0x1f2297e - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n000 -> ret : 0x7eb851b7 - seed : 0x170a40d4
[RANDOM] RtlRandom ** i : 0n001 -> ret : 0x614770d8 - seed : 0x28fa1086
[RANDOM] RtlRandom ** i : 0n002 -> ret : 0x1e6ad652 - seed : 0x5c7cedfb
[RANDOM] RtlRandom ** i : 0n003 -> ret : 0x7f374410 - seed : 0xe1d3692
[RANDOM] RtlRandom ** i : 0n004 -> ret : 0x1f2297e - seed : 0x5cf914e7
[RANDOM] RtlRandom ** i : 0n005 -> ret : 0x767c8778 - seed : 0x2b3e7943
[RANDOM] RtlRandom ** i : 0n006 -> ret : 0x759b7907 - seed : 0x3b117d35
[RANDOM] RtlRandom ** i : 0n007 -> ret : 0x58c53201 - seed : 0x42227ba5
[RANDOM] RtlRandom ** i : 0n008 -> ret : 0x59934e20 - seed : 0x33a48177
[RANDOM] RtlRandom ** i : 0n009 -> ret : 0x5e6ee55e - seed : 0x5c33df1a

Random API in Ring 0

How does these generators really work?

- The most simple one is the *RtlRandom* routine (x86 architecture)
- Taken from *ReactOs* project (it works in a similar manner):

```
/*********************************************************************
 * RtlRandomEx   [NTDLL.@]
 *
 * Generates a random number. This is based on a LCG.
 * This makes it not suitable for Monte Carlo simulations nor
 * cryptographic applications.
 * This one is faster than RtlRandom.
 * See LCG_A, LCG_C, LCG_M for parameters.
 *
 * PARAMS
 *  Seed [O] The seed of the Random function
 *
 * RETURNS
 *  It returns a random number distributed over [0..MAXLONG-1].
 */
/*
 * @implemented
 */
ULONG
NTAPI
RtlRandomEx( IN OUT PULONG Seed
        )
{
   ULONG Rand;
   int Pos;

   PAGED_CODE_RTL();

   Pos = RtlpRandomExAuxVarY & (sizeof(RtlpRandomExConstantVector) / sizeof(RtlpRandomExConstantVector[0]) - 1);
   RtlpRandomExAuxVarY = RtlpRandomExConstantVector[Pos];
   Rand = (*Seed * LCG_A + LCG_C) % LCG_M;
   RtlpRandomExConstantVector[Pos] = Rand;
   *Seed = Rand;

   return Rand;
}
```

```
static ULONG RtlpRandomExAuxVarY = 0x7775fb16;

#define LCG_A 0x7fffffed
#define LCG_C 0x7fffffc3
#define LCG_M MAXLONG
```

```
static ULONG RtlpRandomExConstantVector[128] =
{
    0x4c8bc0aa, 0x4c022957, 0x2232827a, 0x2f1e7626,  /*   0 */
    0x7f8bdafb, 0x5c37d02a, 0x0ab48f72, 0x2f0c4ffa,  /*   4 */
    0x290e1954, 0x6b635f23, 0x5d3885c0, 0x74b49ff8,  /*   8 */
    0x5155fa54, 0x6214ad3f, 0x111e9c29, 0x242a3a09,  /*  12 */
    0x75932ae1, 0x40ac432e, 0x54f7ba7a, 0x585ccbd5,  /*  16 */
    0x6df5c727, 0x0374dad1, 0x7112b3f1, 0x735fc311,  /*  20 */
    0x404331a9, 0x74d97781, 0x64495118, 0x323e04be,  /*  24 */
    0x5974b425, 0x4862e393, 0x62389c1d, 0x28a68b82,  /*  28 */
    0x0f95da37, 0x7a50bbc6, 0x09b0091c, 0x22cdb7b4,  /*  32 */
    0x4faaed26, 0x66417ccd, 0x189e4bfa, 0x1ce4e8dd,  /*  36 */
    0x5274c742, 0x3bdcf4dc, 0x2d94e907, 0x32eac016,  /*  40 */
    0x26d33ca3, 0x60415a8a, 0x31f57880, 0x68c8aa52,  /*  44 */
    0x23eb16da, 0x6204f4a1, 0x373927c1, 0x0d24eb7c,  /*  48 */
    0x06dd7379, 0x2b3be507, 0x0f9c55b1, 0x2c7925eb,  /*  52 */
    0x36d67c9a, 0x42f831d9, 0x5e3961cb, 0x65d637a8,  /*  56 */
    0x24bb3820, 0x4d08e33d, 0x2188754f, 0x147e409e,  /*  60 */
    0x6a9620a0, 0x62e26657, 0x7bd8ce81, 0x11da0abb,  /*  64 */
    0x5f9e7b50, 0x23e444b6, 0x25920c78, 0x5fc894f0,  /*  68 */
    0x5e338cbb, 0x404237fd, 0x1d60f80f, 0x320a1743,  /*  72 */
    0x76013d2b, 0x070294ee, 0x695e243b, 0x56b177fd,  /*  76 */
    0x752492e1, 0x6decd52f, 0x125f5219, 0x139d2e78,  /*  80 */
    0x1898d11e, 0x2f7ee785, 0x4db405d8, 0x1a028a35,  /*  84 */
    0x63f6f323, 0x1f6d0078, 0x307cfd67, 0x3f32a78a,  /*  88 */
    0x6980796c, 0x462b3d83, 0x34b639f2, 0x53fce379,  /*  92 */
    0x74ba50f4, 0x1abc2c4b, 0x5eeaeb8d, 0x335a7a0d,  /*  96 */
    0x3973dd20, 0x0462d66b, 0x159813ff, 0x1e4643fd,  /* 100 */
    0x06bc5c62, 0x3115e3fc, 0x09101613, 0x47af2515,  /* 104 */
    0x4f11ec54, 0x78b99911, 0x3db8dd44, 0x1ec10b9b,  /* 108 */
    0x5b5506ca, 0x773ce092, 0x567be81a, 0x5475b975,  /* 112 */
    0x7a2cde1a, 0x494536f5, 0x34737bb4, 0x76d9750b,  /* 116 */
    0x2a1f6232, 0x2e49644d, 0x7dddcbe7, 0x500cebdb,  /* 120 */
    0x619dab9e, 0x48c626fe, 0x1cda3193, 0x52dabe9d   /* 124 */
};
```

Random API in Ring 0

- *RtlRandom* routine is a weak linear congruential random number generator…

```
v1 = seed;
v2 = (2147483629*v1 + 2147483587) % 0x7FFFFFFFL;
v3 = (2147483629*v2 + 2147483587) % 0x7FFFFFFFL;
v1 = v3;
return(v2);
```

- This type of generator has been (publicly!!) broken many years ago (mid 90s)!
- The *RtlRandomEx* routine is a bit less simplistic (but the two constants are the same)
- From our study in black box, this is more or less the same generator.
  - It uses “variables” initialized and updated by the Windows kernel
  - The constants are contained in *ntoskrnl.exe*
  - It is although a weak linear congruential random number generator…
  - Since we know how extern values evolve, it can easily be broken/predicted
- Conjecture: RtlRandom might be used to produce randomness for processes (salt, Process Color Seed...) or even used *CryptProtectMemory function*.
- Code source available by contacted me
- Analysis under progress… to be continued

Signing Driver Under Windows 7

Signing driver is a key-security point of windows

- If you try to sign a driver under Windows 8 you need a SHA256 certificate
- If you try to sign a driver under Windows 7 you need a SHA1 certificate

From the msdn documentation :

"*In some cases, you might want to sign a driver package with two different signatures. For example, suppose you want your driver to run on Windows 7 and Windows 8. Windows 8 supports signatures created with the SHA256 hashing algorithm, but Windows 7 does not. For Windows 7, you need a signature created with the SHA1 hashing algorithm*".

Question: Why Windows 7 (the most prevalent flavour of windows for the next 10 years) which uses same security mechanism from the kernel) needs to use deprecated technology to sign its drivers ?...

Hypothesis: SHA-1 is fully under control while SHA-256 is not…yet!

Conclusion for this Part

- You do not really need vulnerability when weak architecture design choice exist
- It is obvious that there is a strong will not to provide a high-level security with respect to cryptography mechanism
- Work under progress…to be continued…

The GostCrypt Project

Free Encryption vs NSA

The Issue

- The answer against NSA mass surveillance and eavesdropping is encryption.
- TrueCrypt was until recently the most famous and easy-to-use free solution
  - Certified by the French Prime Minister offices (CSPN)
  - About to be certified by the Open Crypto Audit Project
- In May 2014, TrueCrypt has suddenly disappeared very likely under the pressure of the NSA (no official communication by any of the parties until now)

The GostCrypt Project

- Launched in December 2013, to provide a free, non UKUSA-based alternative to TrueCrypt as a fork of TrueCrypt.
- Use GOST family of cipher and hash functions
  - Contrary to the AES, GOST ciphers have not invaded the world and have always designed by the Russian Federation for its own needs.

GostCrypt Security

- The Sboxes are changed very frequently making any cryptanalysis impossible:
  - The user's secret key modifies the base Sboxes (first level of variation)
  - Every 512-byte ID modifies the Sboxes additionally (second level of variation)
- The algorithm is then changing every 512 bytes (polymorphic algorithm).
- The OS level security has been strenghtened.
- Website: https://www.gostcrypt.org

Conclusion

What will be the future ? How to resist?

Conclusion

- Techniques presented here are classical approaches and are known for a very long time
  - Snowden did not reveal anything really new (at least regarding technical aspects)
  - The issue is: "*what about now and tomorrow?*"
- The strength of the USA is their monopoly over technology (Intel, Microsoft…), services (Facebook, Google, FedEx…) business (in IT, the US market is nearly 45 % of the world market). The real power comes from the commercial power and the political will
- The weight of standards (ISO, ANSI, IEEE) is prevalent. Non US countries must operate at the standardization level or even must create non US-driven standardization organizations AND give birth to a world economic competitor (Europe + Russian Federation)
  - A political issue more than a technical issue

Conclusion

- Cryptography must no longer be the unique solution
  - When you encrypt you send noise! Then you are visible!
- The solution is to generalize the use of steganography especially for network communications.
  - Unsuspected contents cannot be targeted!
- Web browsers become more and more critical software (CAs management)
  - Most countries should develop their own trusted browser
- Hardware must also be taken into account
- Other actors than USA and their allies must also be considered (e.g. China)

Thanks for your attention!
आपका ध्यान के लिए धन्यवाद

Questions & Answers - प्रश्न और उत्तर

References

[1] Eric Filiol "*La face cachée de la sécurité informatique ou les dessous d'Internet*". Club des Vigilants, 14 février 2013 (**in French**). Presented at HIP 2013 and published in Journal in Information Warfare, Vol. 12, Issue 3, October 2013

[2] Peter Wright "*Spycatcher: The Candid Autobiography of a Senior Intelligence Officer", Penguin Viking, 1987.*

[3] *http://emergingtruth.wordpress.com/2013/06/16/is-this-the-early-signs-of-fascism-nsa-cia-fbi-have-secret-agreements-with-private-companies-such-as-microsoft-and-mcafee/*

[4] [6] About Shim. http://blogs.technet.com/b/askperf/archive/2011/06/17/demystifying-shims-or-using-the-app-compat-toolkit-to-make-your-old-stuff-work-with-your-new-stuff.aspx

[5] The GostCrypt Project https://www.gostcrypt.org