Practical Reversing III – Malware Memory Forensics

Monnappa (m0nna)

www.SecurityXploded.com

Disclaimer

The Content, Demonstration, Source Code and Programs presented here is "AS IS" without any warranty or conditions of any kind. Also the views/ideas/knowledge expressed here are solely of the trainer's only and nothing to do with the company or the organization in which the trainer is currently working.

However in no circumstances neither the trainer nor SecurityXploded is responsible for any damage or loss caused due to use or misuse of the information presented here.

Acknowledgement

- Special thanks to **null** & **Garage4Hackers** community for their extended support and cooperation.
- Thanks to all the trainers who have devoted their precious time and countless hours to make it happen.

Reversing & Malware Analysis Training

This presentation is part of our **Reverse Engineering & Malware Analysis** Training program. Currently it is delivered only during our local meet for FREE of cost.

For complete details of this course, visit our Security Training page.

Who am I

Monnappa

- m0nna
- Member of SecurityXploded
- Info Security Investigator @ Cisco
- Reverse Engineering, Malware Analysis, Memory Forensics
- GREM, CEH
- Email: monnappa22@gmail.com

Contents

- Why Memory Forensics?
- Steps in Memory Forensics
- Volatility Quick Overview
- Volatility help and plugins
- Demo

Why Memory Forensics?

- **Finding and extracting forensic artefacts**
- **Helps in malware analysis**
- **Determining process, network, registry activities**
- **Reconstructing original state of the system**
- **Assists with unpacking, rootkit detection and reverse engineering**

Steps in Memory Forensics

- **Memory acquisition - Dumping the memory of a target machine**

  - **tools: Win32dd/Win64dd, Memoryze, DumpIt, FastDump**

  - **In Virtual machine: Suspend the VM and use .vmem file**

- **Memory analysis - Analyzing the memory dump for forensic artifacts**

  - **tools: Volatility, Memoryze**

Volatility Quick Overview

- **Advanced memory Forensics Framework written in python**
- **Installation details:**

  - http://code.google.com/p/volatility/wiki/FullInstallation
- **Use -h or --help option to get list of command-line switches**

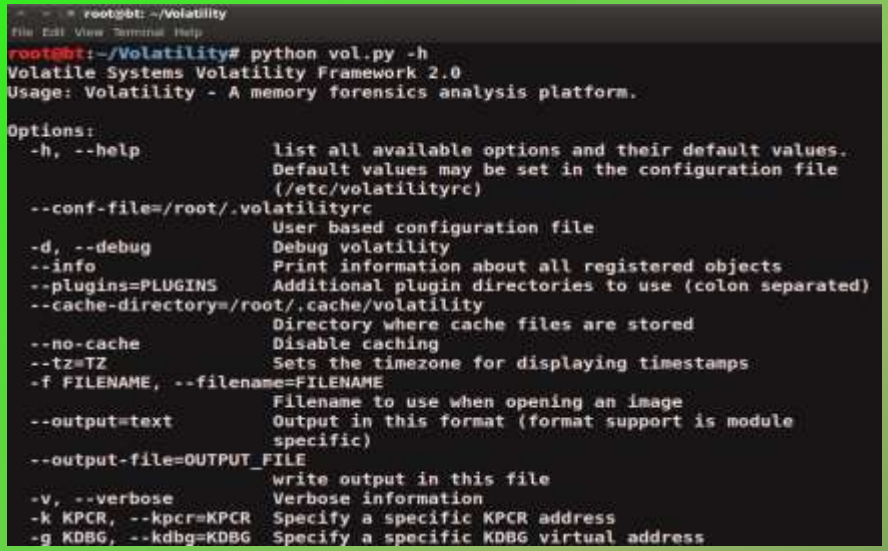
  - example: python vol.py –h
- **Use -f <filename> and --profile to indicate the memory dump you are analyzing**

  example: python vol.py -f mem.dmp --profile=WinXPSP3x86
- **To know the --profile info use below command:**

  example: python vol.py -f mem.dmp imageinfo

Volatility help and plugins

-h or –help option displays help and available plug-in commands in volatility.



DEMO

http://youtu.be/YcVusDjnBxw

Demo-Scenario

Your security device alerts, show malicious http connection to ip address 208.91.197.54 from a source ip 192.168.1.100 on 8th june 2012 at around 13:30hrs...you are asked to investigate and do memory forensics on that machine 192.168.1.100

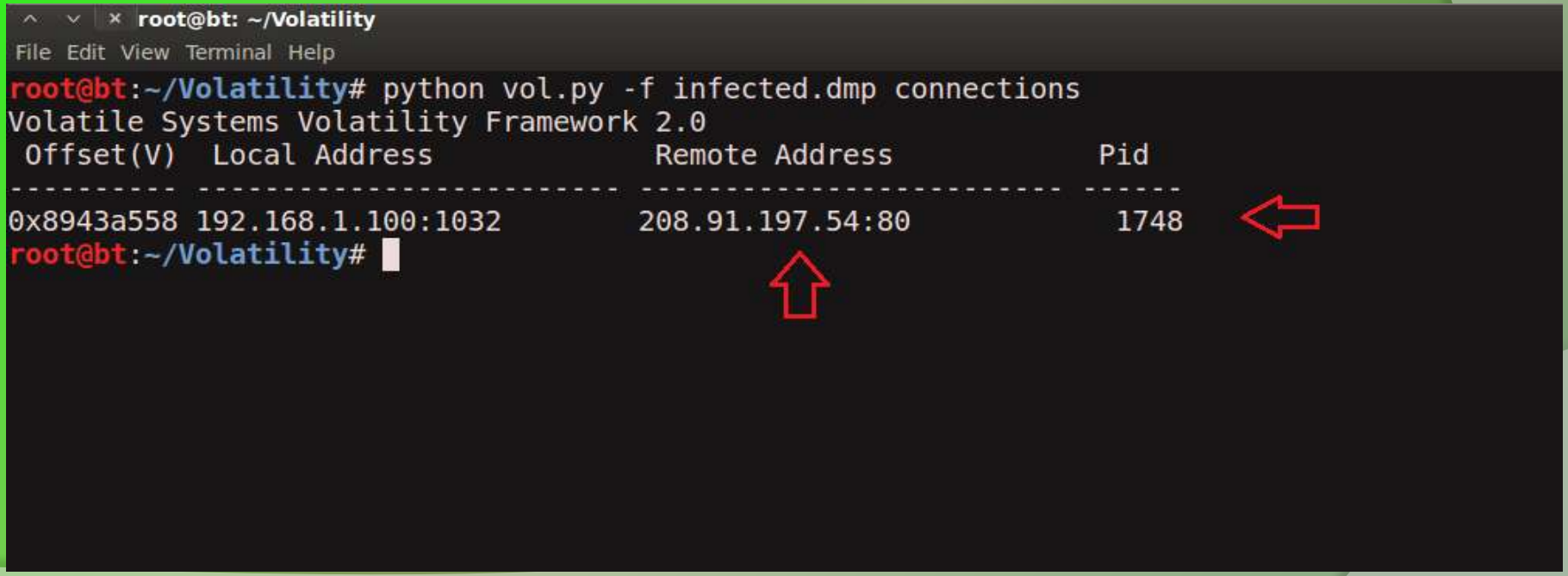
- To start with, acquire the memory image "infected.dmp" from 192.168.1.100, using memory acquistion tools (win32dd)

command: win32dd.exe /f infected.dmp

- Analyze the memory dump "infected.dmp"

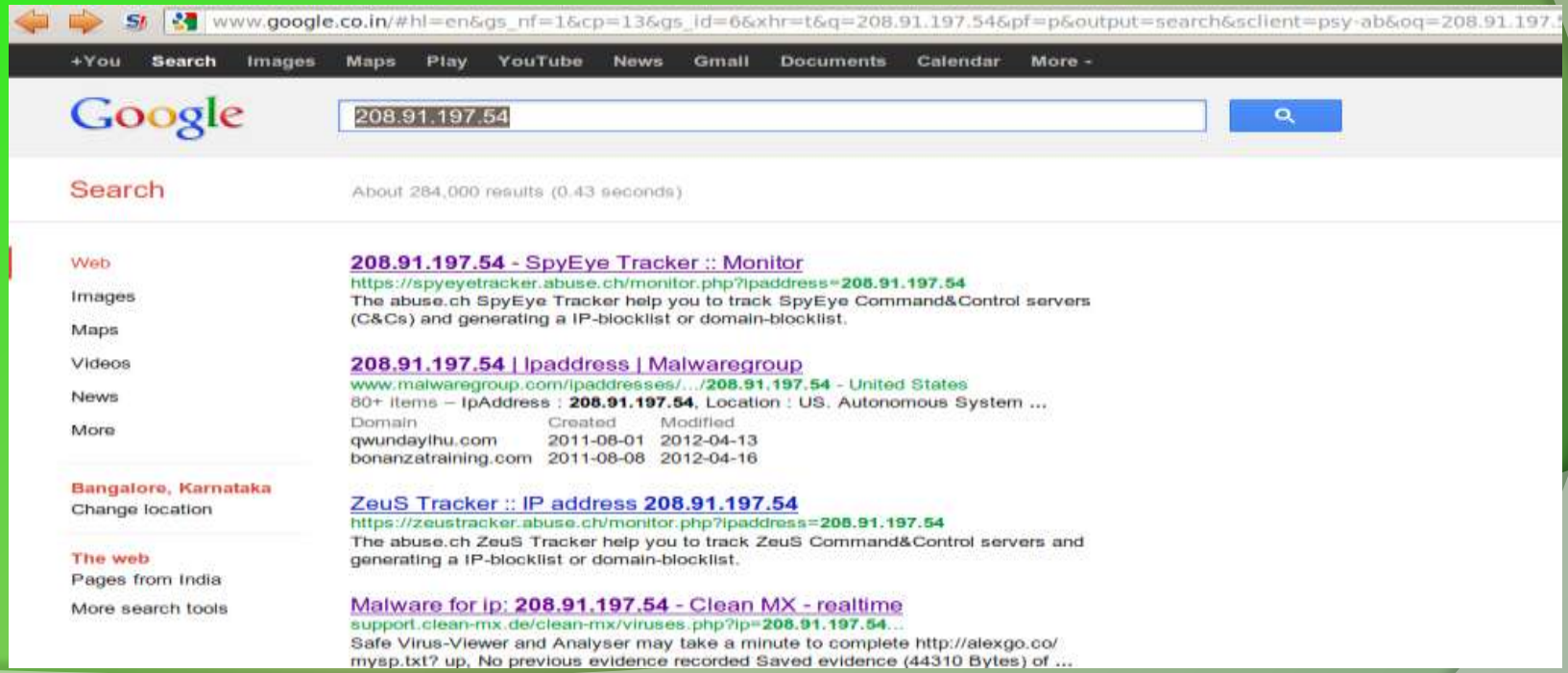
Step 1 – Start With what you know

Volatility's connections module shows connection to the malicious ip by pid 1748



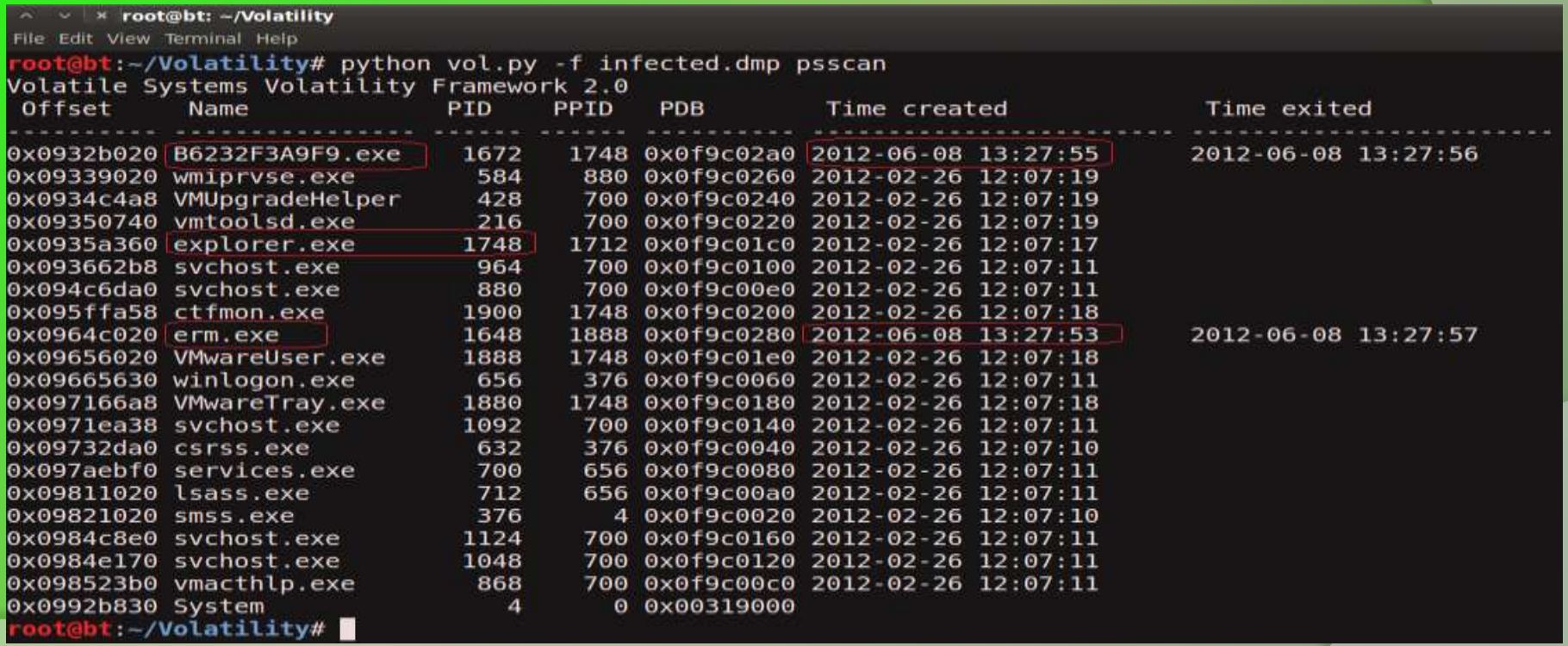
Step 2 – Info about 208.91.197.54

Google search shows 208.91.197.54 associated with malware, probably “spyeye”, we need to confirm that yet.



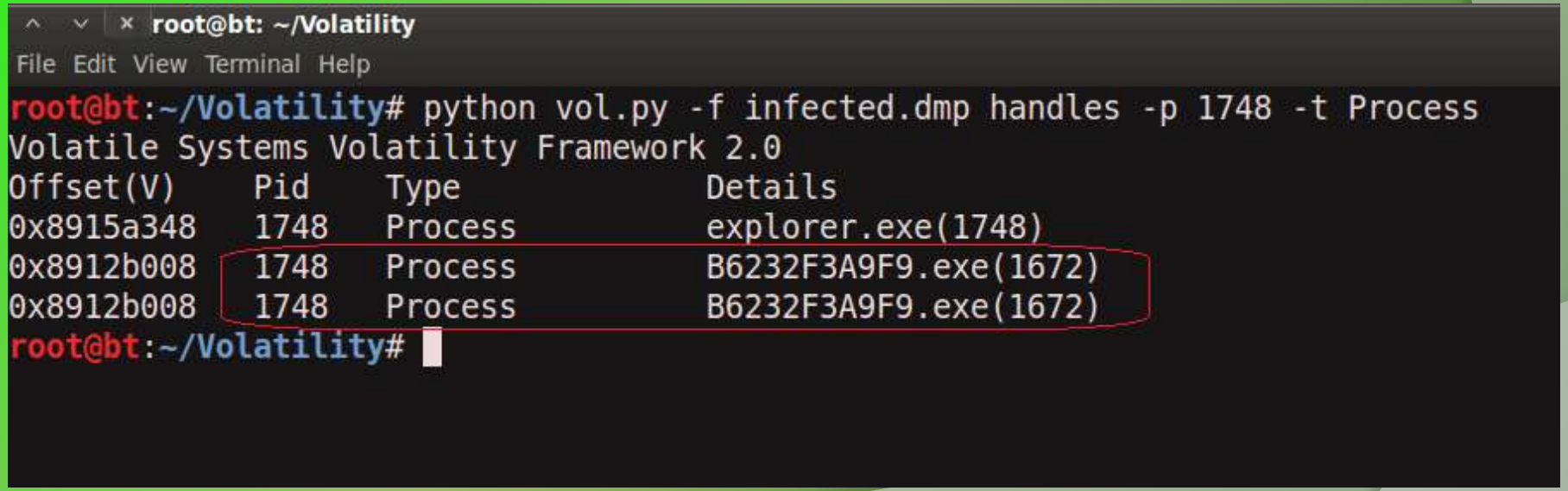
Step 3 – Who is Pid 1748?

"psscan" shows pid 1748 belongs to explorer.exe, also two process created during same time reported by security device (i.e june 8th 2012)



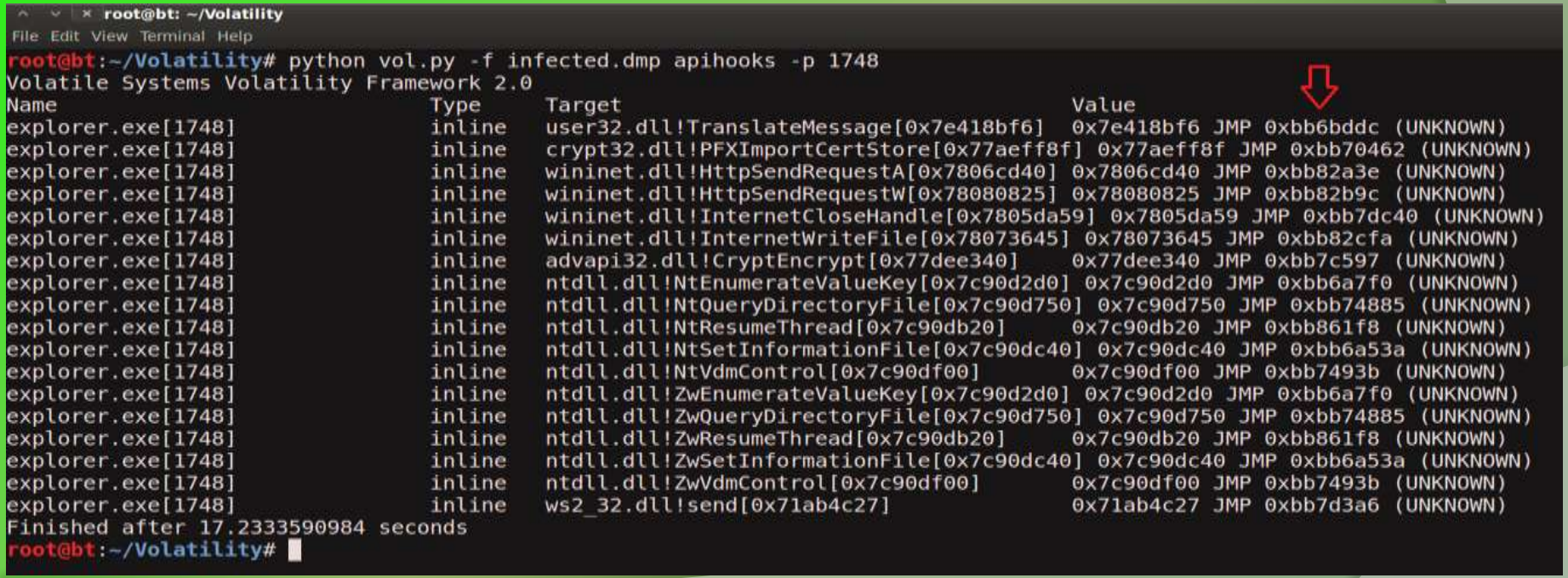
Step 4 – Process handles of explorer.exe

Explorer.exe opens a handle to the B6232F3A9F9.exe, indicating explorer.exe created that process, which might be malicious…focusing on explorer.exe for now.



Step 5 – apihooks in explorer.exe

apihooks module show, inline api hooks in explorer.exe and jump to an unknown location



Step 6 – exploring the hooks

Disassembled hooked function (TranslateMessage), shows a short jump and then a long jump to malware location

```
File  Edit  View  Terminal  Help
root@bt:~/Volatility# python vol.py -f infected.dmp volshell
Volatile Systems Volatility Framework 2.0
Current context: process System, pid=4, ppid=0 DTB=0x319000
Welcome to volshell! Current memory image is:
file:///root/Volatility/infected.dmp
To get help, type 'hh()'
>>> hh()
ps()                                        : Print a process listing.
cc(offset=None, pid=None, name=None)        : Change current shell context.
dd(address, length=128, space=None)         : Print dwords at address.
db(address, length=128, width=16, space=None) : Print bytes as canonical hexdump.
hh(cmd=None)                                : Get help on a command.
dt(objct, address=None)                     : Describe an object or show type info.
list_entry(head, objname, offset=-1, fieldname=None, forward=True) : Traverse a _LIST_ENTRY.
dis(address, length=128, space=None)        : Disassemble code at a given address.

For help on a specific command, type 'hh(<command>)'
>>> cc(pid=1748)
Current context: process explorer.exe, pid=1748, ppid=1712 DTB=0xf9c01c0
>>> dis(0x7e418bf6, length=32)
0x7e418bf6 eb01                                 JMP 0x7e418bf9
0x7e418bf8 c3                                   RET
0x7e418bf9 e9de31758d                           JMP 0xbb6bddc
0x7e418bfe 086681                               OR [ESI-0x7f], AH
0x7e418c01 7e08                                 JLE 0x7e418c0b
0x7e418c03 e500                                 IN EAX, 0x0
0x7e418c05 0f84667e0200                         JZ 0x7e440a71
0x7e418c0b 6a00                                 PUSH 0x0
```

Step 7 – Embedded exe in explorer.exe

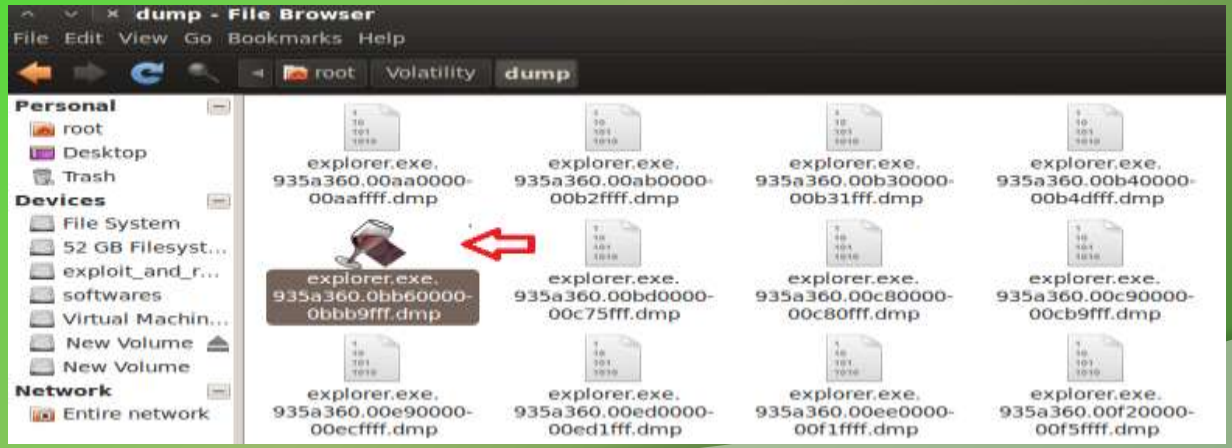
Printing the bytes show the presence of embedded executable in explorer.exe

```
>>> db(0x0bb60000, length=256)
0bb60000   4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00    MZ..............
0bb60010   b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00    ........@.......
0bb60020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb60030   00 00 00 00 00 00 00 00 00 00 00 00 e0 00 00 00    ................
0bb60040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb60050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb60060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb60070   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb60080   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb60090   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb600a0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb600b0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb600c0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb600d0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0bb600e0   50 45 00 00 4c 01 02 00 92 60 ed 4d 00 00 00 00    PE..L....`.M....
0bb600f0   00 00 00 00 e0 00 02 01 0b 01 0a 00 00 a2 04 00    ................

>>>
```

Step 8 – dumping the embedded exe

vaddump dumps the embedded exe from explorer.exe

```
root@bt: ~/Volatility
File  Edit  View  Terminal  Help
root@bt:~/Volatility# python vol.py -f infected.dmp vaddump -p 1748 -D dump/
Volatile Systems Volatility Framework 2.0
Pid:    1748
************************************************************************
root@bt:~/Volatility#
```



Step 9 – virustotal submission

Submission to virustotal, confirms the dumped executable as component of “spyeye”

Detection ratio: 8 / 39

Analysis date: 2012-06-08 19:56:31 UTC (2 minutes ago)

More details

Antivirus	Result	Update
AhnLab-V3	Packed/Win32.Morphine	20120608
AntiVir	TR/Dropper.Gen	20120608
Antiy-AVL	-	20120608
Avast	Win32:Spyeye-XY [Trj]	20120608
BitDefender	-	20120608
ByteHero	-	20120606
CAT-QuickHeal	-	20120608
ClamAV	-	20120608
Commtouch	-	20120608
Comodo	-	20120608
Emsisoft	Trojan.Win32.SpyeyeIIK	20120608
eSafe	-	20120607
F-Prot	-	20120608
F-Secure	-	20120608
Fortinet	-	20120608
GData	Win32:Spyeye-XY	20120608
Ikarus	Trojan.Win32.Spyeye	20120608

Step 10 – Can we get more info?

Strings extracted from the dumped executable, show reference to interesting artifacts (executable and the registry key)

```
root@bt: ~/Volatility/test
File  Edit  View  Terminal  Help
root@bt:~/Volatility/test# strings explorer.exe.935a360.0bb60000-0bbb9fff.dmp > ascii strings.txt
```

```
Connection: close
Connection:
Cteonnt-Length:
Content-Length:
Content-Encoding:
Content-Encoding: deflate
Content-Encoding: gzip
Transfer-Encoding:
chunked
Content-Length: %u
HTTP/
User-Agent:
Accept-Encoding:
Keep-Alive:
Connection: keep-alive
Proxy-Connection: keep-alive
SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN
\*.*
%.2x
cookies-nontor.xml
cookies.txt
sessionstore.js
sessionstore.bak
```

```
f98u
_^[t
8&Ol
C:\WINDOWS\system32\WININET.dll
C:\Recycle.Bin\A705B3960358085
C:\WINDOWS\system32\ntdll.dll
C:\WINDOWS\system32\ntdll.dll
C:\WINDOWS\system32\USER32.dll
C:\WINDOWS\system32\ntdll.dll
C:\WINDOWS\system32\CRYPT32.dll
C:\Recycle.Bin\B6232F3A9F9.exe
C:\Recycle.Bin\
B6232F3A9F9.exe
A705B3960358085
s1PSg1LF.exe
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\
```

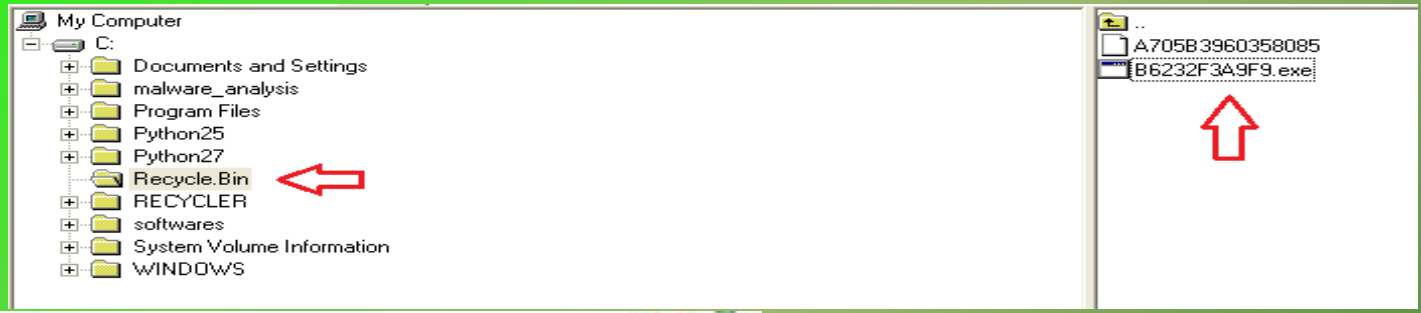
Step 11 – Printing the registry key

Malware creates registry key to survive the reboot

```
root@bt:~/Volatility# python vol.py -f infected.dmp printkey -K "SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN"
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
Key name: Run (S)
Last updated: 2011-10-31 15:07:20

Subkeys:

Values:
----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\default
Key name: Run (S)
Last updated: 2011-10-31 20:28:57

Subkeys:

Values:
----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT
Key name: Run (S)
Last updated: 2012-06-08 13:27:56

Subkeys:

Values:
REG_SZ        ctfmon.exe      : (S) C:\WINDOWS\system32\ctfmon.exe
REG_SZ        4Y3Y0C3A1F7XZHZWACQCUD : (S) C:\Recycle.Bin\B6232F3A9F9.exe
```

Step 12 – Finding the malicious exe on infected machine

Finding malicious sample from infected host and virustotal submission confirms spyeye infection



Antivirus	Result	Update
AhnLab-V3	Spyware/Win32.SpyEyes	20120608
AntiVir	TR/EyeStye.N.2112	20120608
Antiy-AVL	Trojan/Win32.SpyEyes.gen	20120608
Avast	Win32:Spyeye-XY [Trj]	20120608
AVG	PSW.Generic8.CAPI	20120608
BitDefender	Trojan.Agent.ASCZ	20120608
Bytehero	-	20120608
CAT-QuickHeal	TrojanSpy.SpyEyes.ikd	20120608
ClamAV	PUA.Win32.Packer.Anti-4	20120608

Kaspersky	Trojan-Spy.Win32.SpyEyes.ikd
McAfee	PWS-Spyeye.q
McAfee-GW-Edition	Heuristic.BehavesLike.Win32.ModifiedUPX.C
Microsoft	Trojan:Win32/EyeStye.N
NOD32	a variant of Win32/Spy.SpyEye.CA
Norman	W32/Suspicious_Gen2.QPFLH
nProtect	Trojan/W32.Agent.320512.CW
Panda	-
PCTools	Trojan.Spyeye

Reference

- Complete Reference Guide for Reversing & Malware Analysis Training

Thank You !

www.SecurityXploded.com