GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Reverse-Engineering Malware: Malware Analysis Tools and Techniques (Forens
at http://www.giac.org/registration/grem



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Reverse-Engineering Malware: Malware Analysis Tools and Techniques (Foren
at http://www.giac.org/registration/grem

A Detailed Analysis of an Advanced Persistent Threat Malware

GIAC (GREM) Gold Certification

Author: Frankie Li, ran2@vxrl.org
Advisor: Antonios Atlasis

Accepted: October 13, 2011

Abstract

Spear-phishing emails were sent to a political figure at my place of residence. An email together with the attached sample was provided for forensics analysis. It appears to be an Advanced Persistent Threat type malware. By performing behavioral and code analysis in an alternatively way, most of its important functions were identified. The aim of this technical paper is to illustrate the detailed procedures of how this malware was dissected.

© 2011 SANS Institute, Author retains full rights.

© 2011 The SANS Institute Author retains full rights.

1. Introduction

Spear-phishing emails were sent to a political figure at my place of residence. An email, including the attached sample was provided for forensics analysis. This email contained obviously well crafted message to lure the recipient to open the malicious attachment. It was predicted as an Advanced Persistent Threat attack (APT-attack).

By performing the detailed behavioral and code analysis in a Spiral way (Brand, Valli & Woodward, 2010, p 6), most of the important functions of the malware were identified. The aim of this technical paper is to demonstrate the step-by-step procedures on how this malware was dissected.

Advance Persistent Threat (APT) is a hot and controversial term used amongst security professionals, including Bejtlich (2011), Cloppert (2010), Lee (2011) and Hoglund (2011), especially after McAfee published their white paper called: "Revealed: Operation Shady RAT" (Alperovitch, 2011). Kaspersky criticized this report and flagged "the report as alarmist due to its deliberately spreading misrepresented information" (Frye, 2011).

The term APT is frequently used as a replacement term to describe cyber warfare between countries (Cloppert, 2009). Albert Hui, an IT professional in Hong Kong viewed it as "an entire threat class (analogous to other threat classes like insider fraud, industrial espionage, and hactivism), whereas things like RATs, drive-by malwares, rootkits, DDoS are threat vehicles" (Hui, 2011). Referring to the traditional meaning of APT, the malware is only one of the threat components of APT. However, most anti-virus vendors, including McAfee and Kaspersky continue to fame APT as a malware. One side of them even treats it as a "bot" or "back door", which contains sophisticated techniques that persistently hiding itself in the victim's system for collection of intelligence. However, comparing with other threat tools, because malware is easier to prepare and deploy, the term APT will be continuously be described as a malware. A fine-tuned definition proclaimed by Mandiant (2011, "What is M-Trends?") is accepted. In this technical paper, APT is considered as a cyber attack launched by a group of sophisticated, determined and coordinated attackers that have been systematically

© 2011 The SANS Institute Author retains full rights.

compromising a specific target's machine or entity's networks for prolonged period. The meaning of "Persistence" is also expanded to the acts of the attackers of persistently launching spear-phishing attacks against the targets. The findings indicate the main functions of the APT-type malware are usually placed heavily on spying instead of for the purpose of financial gain.

Other than the purpose of collecting of national secrets or political espionage, based on the functions discovered, it is believed that this threat can also apply to the cases in business or industrial espionages, spying acts or even un-ethical detective investigations.

2. The Behavioral and Code Analysis

2.1. Setting Up the Lab Environment

The analysis was conducted inside a host-only network VMware machines.

a) The infection box was installed with a Windows XP SP2 system. The IP address was configured as 192.168.80.125 with default gateway and DNS pointed to 192.168.80.130 (i.e. IP address of the responsive box described below).

b) A lightweight Ubuntu distribution, REMnux 2.0 was used for interacting with the malware. It was configured with IP address of 192.168.80.130 as a responsive box.

The analysis methodology proposed by Zeltser (2007) was adopted. It is a way of molding of analysis environment that alternately uses behavioral and code analysis techniques to identify the functionality of the executable (Valli & Brand, 2008, p.2).

The table shows the tools that were used to perform the behavioral and code analysis (Table 1).

© 2011 The SANS Institute Author retains full rights.

Analysis Technique	Tool name	Reference	Functional usage of the tool
Behavioral	Autoruns (v10.06)	Russinovich, M. (2011a)	List auto-start locations.
Behavioral	Process Explorer (v2.93)	Russinovich, M. (2011b)	Display processes, threads, DLLs loaded.
Behavioral	Process Monitor (v15.0)	Russinovich, M. (2011c)	Log files, registry, network, processes, threads changes.
Behavioral	ListDLLs (v3.1)	Russinovich, M. (2011d)	Display DLLs loaded on the system.
Behavioral	TCPView (v3.02)	Russinovich, M. (2011e)	Lists active TCP/UDP endpoints.
Behavioral	VMmap (v3.1)	Russinovich, M. (2011f)	Display of a process' virtual and physical memory usage.
Behavioral	Winobj (v.2.21)	Russinovich, M. (2011g)	Display Windows' Object Manager namespace.
Behavioral	BinText (v3.00)	Founstone. (2000)	Text extractor.
Behavioral	Regshot (v1.8.2)	Buecher, M., TiANWEi & XhmikosR (2007)	Shows registry and file changes between two of its snapshots.
Behavioral	CaptureBAT (v2.0.0)	Zealand Honeynet Project (2007)	A client honeypot.
Behavioral	HandleDiff (v.0.2)	Ligh, M., Adair, S., Hartstein, B., & Richard, M. (2011b)	Detect changes to the handle tables of process.
Behavioral	Wireshark (v1.2.2)	Wireshark.org. (2010)	Network Protocol Analyzer and packet capture utility.
Behavioral	Malcode Analysis Pack (v1.0)	Zimmer, D. (2001)	A package contains applications that have proven useful for analyzing malicious code.
Behavioral/Code	REMnux (v2.0)	Zeltser, L. (2011)	A Lightweight Linux distribution for assisting malware analysts in reverse-engineering malicious software.
Code	UPX (v3.0.3w)	Oberhumer, M. (2008)	A tool that can achieves and de-achieve an executable.
Code	FileInsight (v2.1)	McAfee (2009)	a file reader that displays the document in either textual or hexadecimal format.
Code	OllyDbg (OllyICE Lite) (v1.10)	Yuschuk, O. (2004)	Debugger for Win32 binaries.
Code	IDA Pro Freeware (v5.0)	Hex-Rays SA. (2010)	Disassembler and debugger with graphing capabilities.
Code	PEiD (v0.94)	Snaker, Qwerton & Jibz (2008)	PE File identification tool.
Code	Stud PE (v2.4.0.1)	Gheorghe, C. (2008)	PE format Viewer.

Table 1. Tools used to perform behavioral and code analysis

© 2011 The SANS Institute Author retains full rights.

2.2. The Observed Behavior

The sample was contained inside a well-crafted phishing email, which described the news of a riot incident happened at a county near the Guangzhou City in China. The malicious attachment named: "XinTang Event.rar". It was not password protected and after decompressed, a file called "XinTang Event.chm" was created. If double-clicked, the following message was displayed (Figure 1).

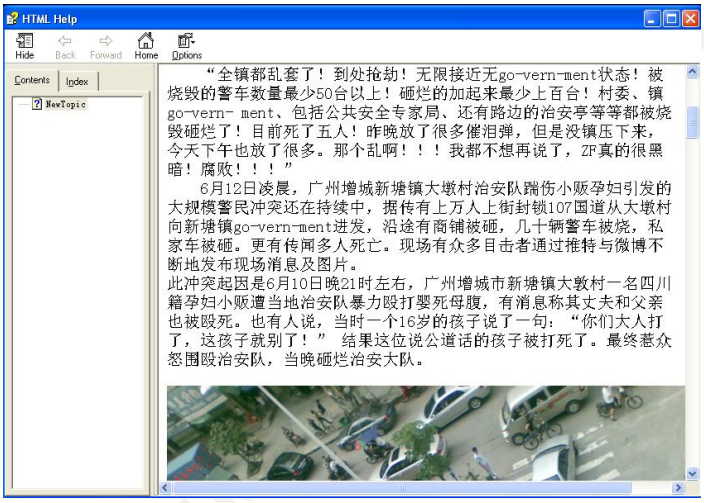


Figure 1. The displayed chm help file if double clicked

To clear up the possible infection and assuming this malware did not exploit any known or 0-day VMware vulnerability, the snapshot of the infection box was rolled-back and uploaded the decompressed chm file to the responsive box for checking with the "file" utility. It was identified as a "MS Windows HtmlHelp Data file". After Googling, an instruction on how to create Trojan horse using chm file was found from a Chinese website (Chen Yi-Tian, 2005)

The chm file was decompiled by using a tool call "Malcode Analysis Pack". A folder (Figure 2) called "chm_src" was generated containing some jpeg files and an executable called "dg003_improve_8080_V132.exe" (hereafter also called as dg003.exe).

© 2011 The SANS Institute Author retains full rights.

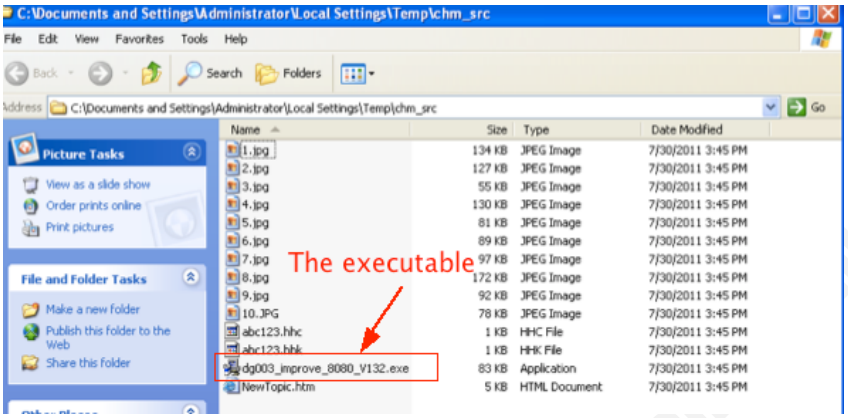


Figure 2. The contents of the decompiled chm_src folder

The file "dg003.exe" was checked with PEiD v0.94 and found UPX-packed (Figure 3). After unpacked using the UPX utility, the file size of "dg003.exe" was expanded from 84,992 bytes to 196,608 bytes generating hash value of 4EC0027BEF4D7E1786A04D021FA8A67F

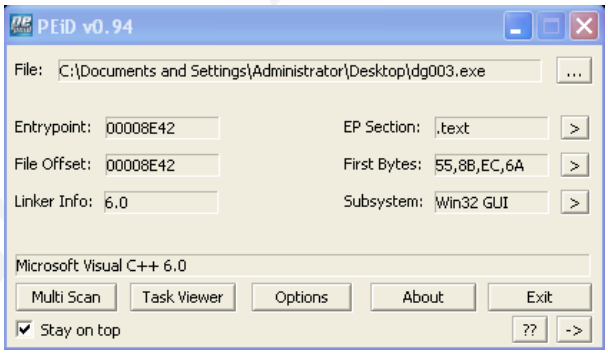


Figure 3. The file "dg003.exe" was checked with PEiD

Various monitoring tools, including: Autoruns, TCPView, Process Explorer, Regshot, Process Monitor, CaputreBAT and HandleDiff were activated in an order to capture maximum information during execution of the malicious "dg003.exe".

The CaptureBAT log was imported into Microsoft Excel for quick review (Li, 2011). Using the smart-filtering feature, all modifications of the registry entries and file system were identified (Figure 4).

Frankie Li, ran2@vxrl.org

© 2011 The SANS Institute Author retains full rights.

	42:28	proce	created	C:\WINDOWS\system32\cmd.exe	C:\Tools\HandleDiff.exe
72	42:48.1	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\ws2help.PNF
73	42:48.1	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\ws2help.PNF
74	42:48.1	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\ws2help.PNF
75	42:48.1	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\ws2help.PNF
76	42:48.1	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\ws2help.PNF
77	42:48.1	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\ws2help.PNF
78	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
79	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
80	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
81	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
82	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
83	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
84	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
85	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
86	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
87	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
88	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
89	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
90	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
91	42:48.2	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\msvcr.dll
92	42:48.3	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\WINDOWS\inf\1.txt
93	42:48.3	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\WINDOWS\inf\1.txt
95	42:48.3	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\WINDOWS\inf\1.txt
101	42:48.4	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\WINDOWS\system32\netstat.exe
102	42:48.5	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\IECheck.exe
103	42:48.5	file	Write	C:\WINDOWS\Downloaded Program Files\dg003_improve_8080_V132.exe	C:\Documents and Settings\Administrator\Local Settings\Application Data\IECheck.exe
1233					

Figure 4. Smart-filtering feature was used for quick identification of modifications

CaptureBAT log was first checked because of its distinctive advantages in behavioral analysis. It can (a) identify which process is responsible for the file or registry modifications; (b) backup the modified or deleted files into a folder during execution; and (c) create a .pcap file to log the network activities during the capture. However, if files were in accessed during the capture, CaptureBAT would miss backing up of these files. As a supplement, Regshot was also activated to capture similar changes.

The file system and registry changes are summarized as below (Table 2).

Tool	Process	Location	Event	Name
CaptureBAT /Regshot	dg003.exe	C:\Documents and Settings\<user>\Local Settings\Application\Data	File added	ws2help.PNF
CaptureBAT /Regshot	dg003.exe	C:\Documents and Settings\<user>\Local Settings\Application\Data	File added	msvcr.dll
CaptureBAT	dg003.exe	C:\Windows\system32	File altered	netstat.exe
CaptureBAT	dg003.exe	C:\Windows\inf	File added & removed	1.txt
CaptureBAT /Regshot	dg003.exe	C:\Documents and Settings\<user>\Local Settings\Application\Data	File added	IECheck.exe
CaptureBAT /Regshot	explorer.exe	C:\Windows\system32	File added	Ipsecstap.dat
CaptureBAT	explorer.exe	C:\Documents and Settings\<user>\Start Menu\Programs\Startup	File added	Internet Explorer Security Check.lnk
Regshot		C:\Windows\system32	File added	13605
HandleDiff	explorer.exe	\BaseNamedObjects	Mutant added	VistaDLLPro RUNNING

Table 2. Changes on file system and registry

Frankie Li, ran2@vxrl.org

© 2011 The SANS Institute Author retains full rights.

Using the tool – Vmmap and ListDLLs, the malicious dynamic-link library (DLL) - "msvcr.dll" was found as an injected thread in the running process of "explorer.exe" at the base address of 0x10000000 with image size of 0x43000 (Figure 5).

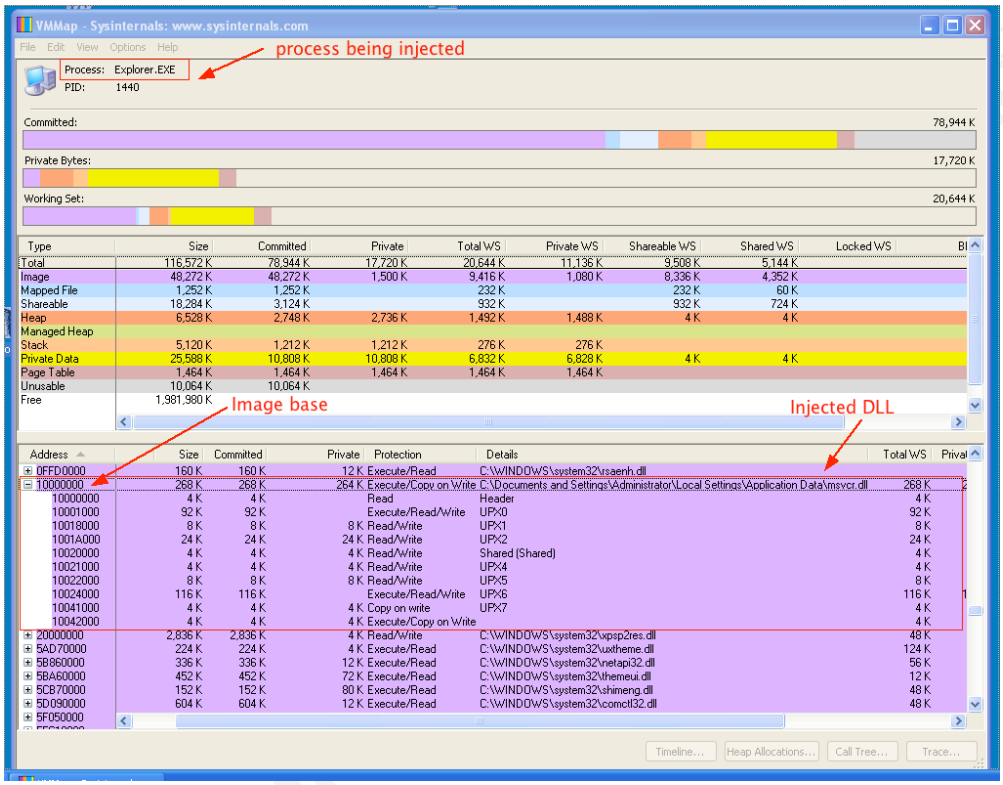


Figure 5. Injected DLL was found by using Vmmap

Checking the Thread information of the injected process - "explorer.exe" with Process Explorer, three additional threads of "msvcr.dll" were discovered and they were started at relative virtual address (RVA) of 0xAEE0, 0xABB0 and 0xA940 (Figure 6).

Wireshark was used to inspect a file named "192.168.80.125.pcap", which was generated by CaptureBAT. The infection box tried to resolve the non-existent hostnames of "test.3322.org.cn", "1.test.3322.org.cn", "2.test.3322.org.cn", "3.test.3322.org.cn" and "4.test.3322.org.cn". Immediate after the unsuccessful retrials, it tried to connect to IP addresses 172.16.0.61 and 115.x.x.249 by using TCP port number 8080.

Frankie Li, ran2@vxrl.org

© 2011 The SANS Institute Author retains full rights.

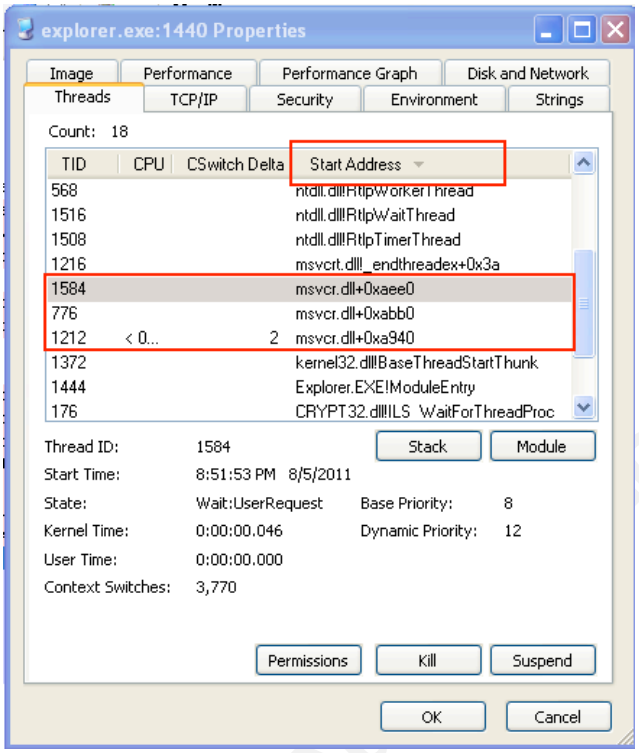


Figure 6. Three additional threads was indentied by using Process Explorer

In order to allow "dg003.exe" to establish connections with these IP addresses inside a controlled virtual environment, the honeyd and farpd services on the Remnux responsive box were activated. The farpd was configured to reply to any ARP request for an IP address matching with the MAC address of the Remnux network adaptor and the honeyd was configured to run http service at TCP port 8080 on the Remnux responsive box (Appendix I). After these services started, some encrypted http communicated was captured and the detail discussion of this communication can be found at section 3.2 of this paper.

3. Technical details discovered by Code Analysis

The malware was identified as a kind of "multi-staged" malware (i.e. a malware using form of multipartite infection strategy) (Szor, 2005, p. 76). During the analysis, which will be described below, the malware or the file "dg003.exe" was found as a dropper, which dropped the file "msvcr.dll" as the droppee. If the Internet access is enabled, the injected "msvcr.dll" will download some more Trojan-Spies.

Frankie Li, ran2@vxrl.org

© 2011 The SANS Institute Author retains full rights.

IDA Pro was used for disassembling the binaries. The graphical view was used along with the "Chart of X-ref to" and "Chart of X-ref from" functions to identify the program flows between each important subroutine.

If possible, every subroutine was dynamically studied by stepping through under a debugger, OllyDbg. Whenever a key function was identified, the subroutine was renamed with a meaningful name both in OllyDbg and IDA Pro. If variables or pass though arguments were found during debugging with OllyDbg, appropriate comments were added to OllyDbg and IDA Pro. By switching back and forth, pseudo-code flowcharts for all investigating binary could be generated.

3.1. The Code Analysis of "dg003.exe"

For quick identification, the file "dg003.exe" was uploaded to VirusTotal. The report indicated only 15 out of 44 anti-viurs engines could detect it as malicious, but not a single one could clearly identify its virus family or signature.

The following diagram shows a board overview of the order of executions and key functions of the file "dg003.exe" (Figure 7).

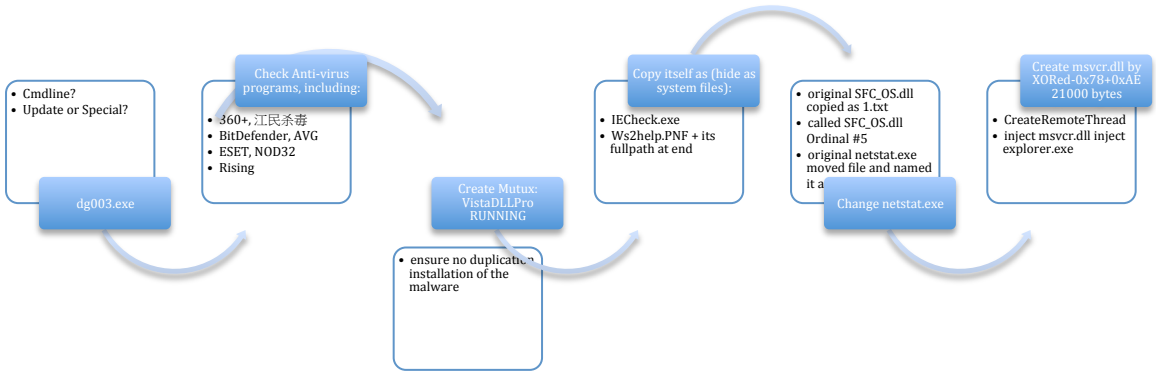


Figure 7. The key functions of "dg003.exe"

The file "dg003.exe" starts by checking if it is called from a command prompt with passing argument of "Update" or "Special" (Figure 8). These parameters were passed to OllyDbg as argument during different debugging sessions. However, no obvious function was triggered.

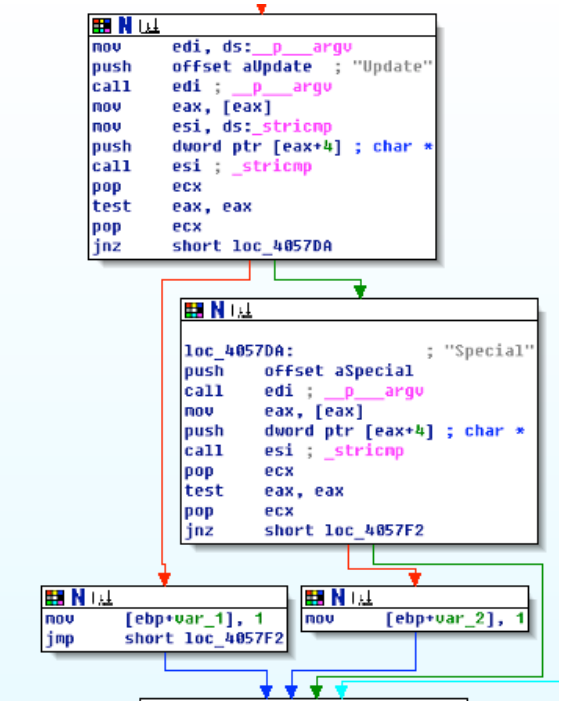


Figure 8. Argument checking was found under IDA Pro

Then, it checks if the victim is installed with some anti-virus programs of ”Kaspersky”, “ESET”, “BitDefender”, “AVG”, “NOD32”, “Rising” or “360+” by enumerating the registry key at “SOFTWARE\Microsoft\windows\CurrentVersion\ Uninstall”

After checking with the passing arguments, “dg003.exe” tries to create a mutux named “VistaDLLPro RUNNING” (Figure 9) to prevent double installation of itself on the system. This malware uses the similar method like, ZeuS bot to mark its presence on the system (Ligh, Adair, Hartstein & Richard, 2011a. p.301).

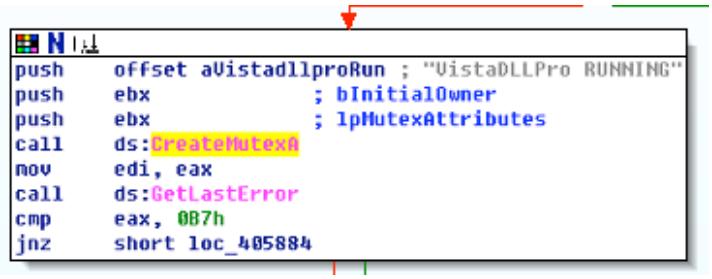


Figure 9. CreateMutux was found under IDA Pro

Then, “dg003.exe” writes a duplicate as C:\Documents and Settings\<user>\Local Settings\Application Data\w2help.PNF at 0x00403DA2 (Figure 10).

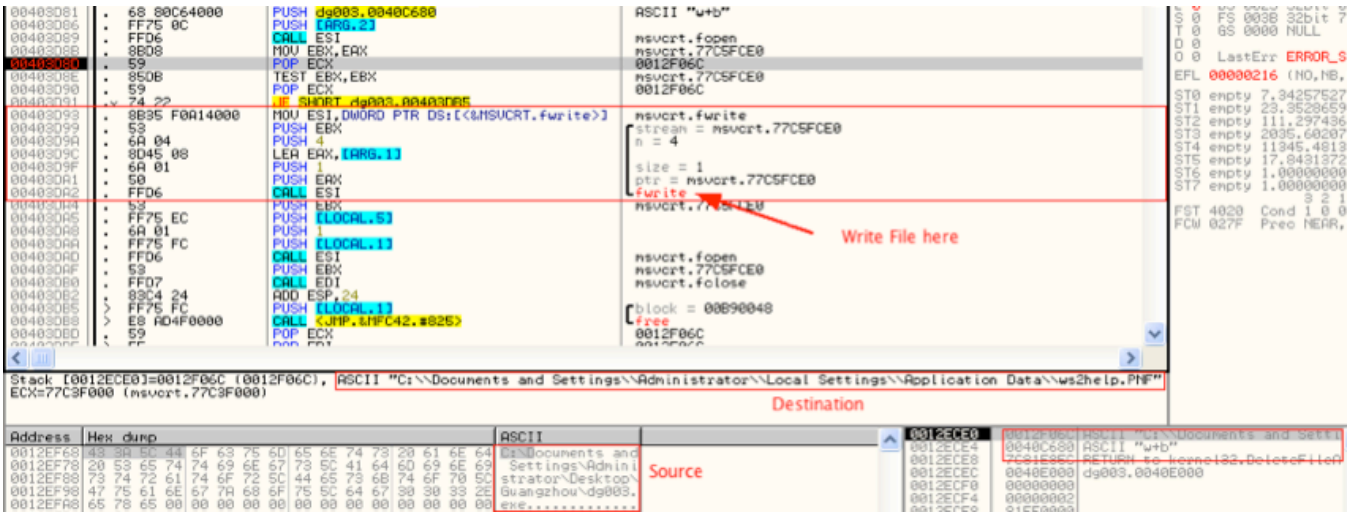


Figure 10. The file “w2help.PNF” was written

Then it calls into 0x00402BB9 to append string of “C:\Documents and Settings\<user>\ Local Settings\Application Data\msvcr.dll” at the end of the file “w2help.PNF” to prevent detection of checksum based detection. At 0x00402BC9, “dg003.exe” creates a DLL in the memory from the code stored at the resource section with ID node name “VISTADLL” (Figure 11).

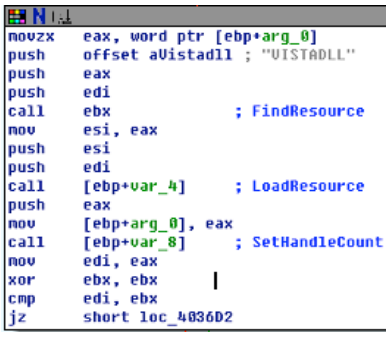


Figure 11. ID node “VISTALDLL” was found under IDA Pro

At 0x0040367F, it copies 0x21000 (135,158) bytes from the resource section to the memory address at 0x0040F1F0. After executing the decoding routine at 0x0043691 to 0x004036AB, a DLL is decoded at memory location of 0x0040F1F0 (Figure 12).

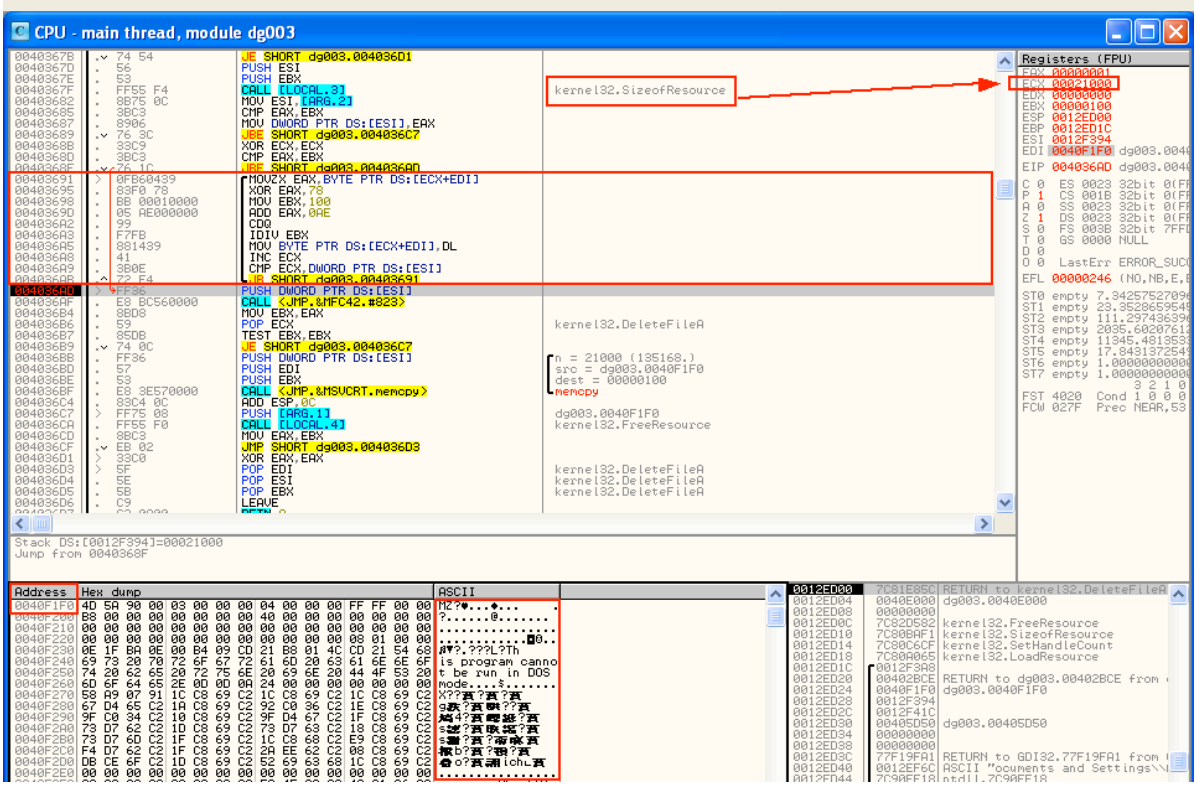


Figure 12. The Decoding Routine

Subsequent the decoding, the DLL is packed with a proprietary packing stub in the memory and "dg003.exe" writes the packed DLL in name of "msvcr.dll" at 0x00402C71 (Figure 13).

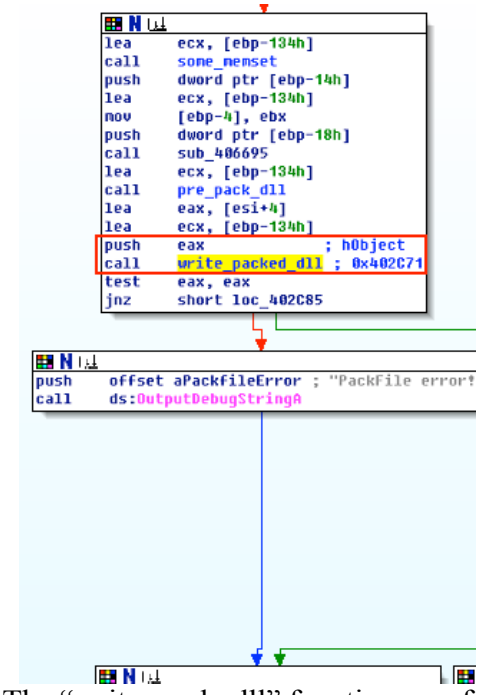


Figure 13. The "write_pack_dll" function was found under IDA Pro

After some clean ups, “dg003.exe” changes the MAC time (i.e. the Modification time, Access time and Change time) of the newly created file “msvcr.dll” at 0x00402D15 and hides “msvcr.dll” as system file at 0x00402D41.

At 0x004017A2, “dg003.exe” copies two Windows system files of “netstat.exe” to C:\Windows\System32\13605 and “SFC_OS.dll” to C:\Windows\inf\1.txt. The “SFC_OS.dll” is the executable portion of Windows File Protection mechanism (WFP), which protects system files from being modified or deleted. The malware calls to ordinal 5 function of “1.txt” in order to bypass the WFP (Collake, 2006) during patching "netstat.exe" for hiding network connection of IP address of 115.x.x.249 (Figure 14).

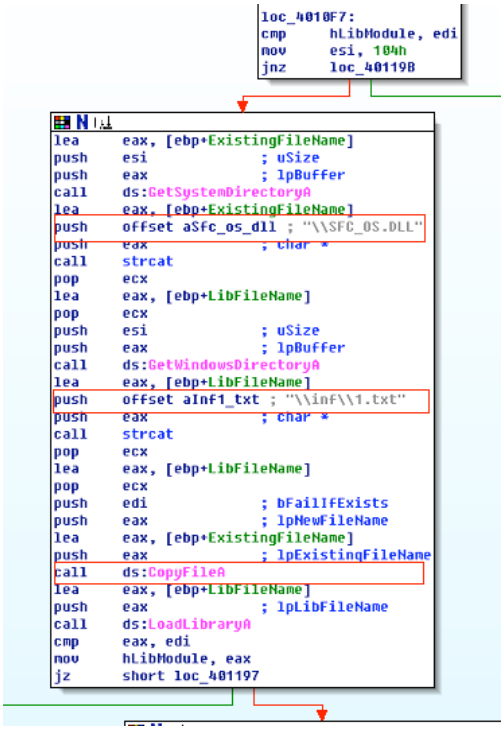


Figure 14. Copy and patch “SFC_OS.dll”

During the behavioral analysis, it was found that “dg003.exe” would inject “msvcr.dll” into the running “explorer.exe” process and because the process is the GUI shell of Windows system, simply attaching this process to OllyDbg will freeze the system. To execute two copies of “explorer.exe”, a Dword of DesktopProcess=1 was added to the registry of HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer. Then, the second copy of C:\WINDOWS\explorer.exe was attached under OllyDbg (hereafter it is called as debugging “explorer.exe”) (Figure 15).

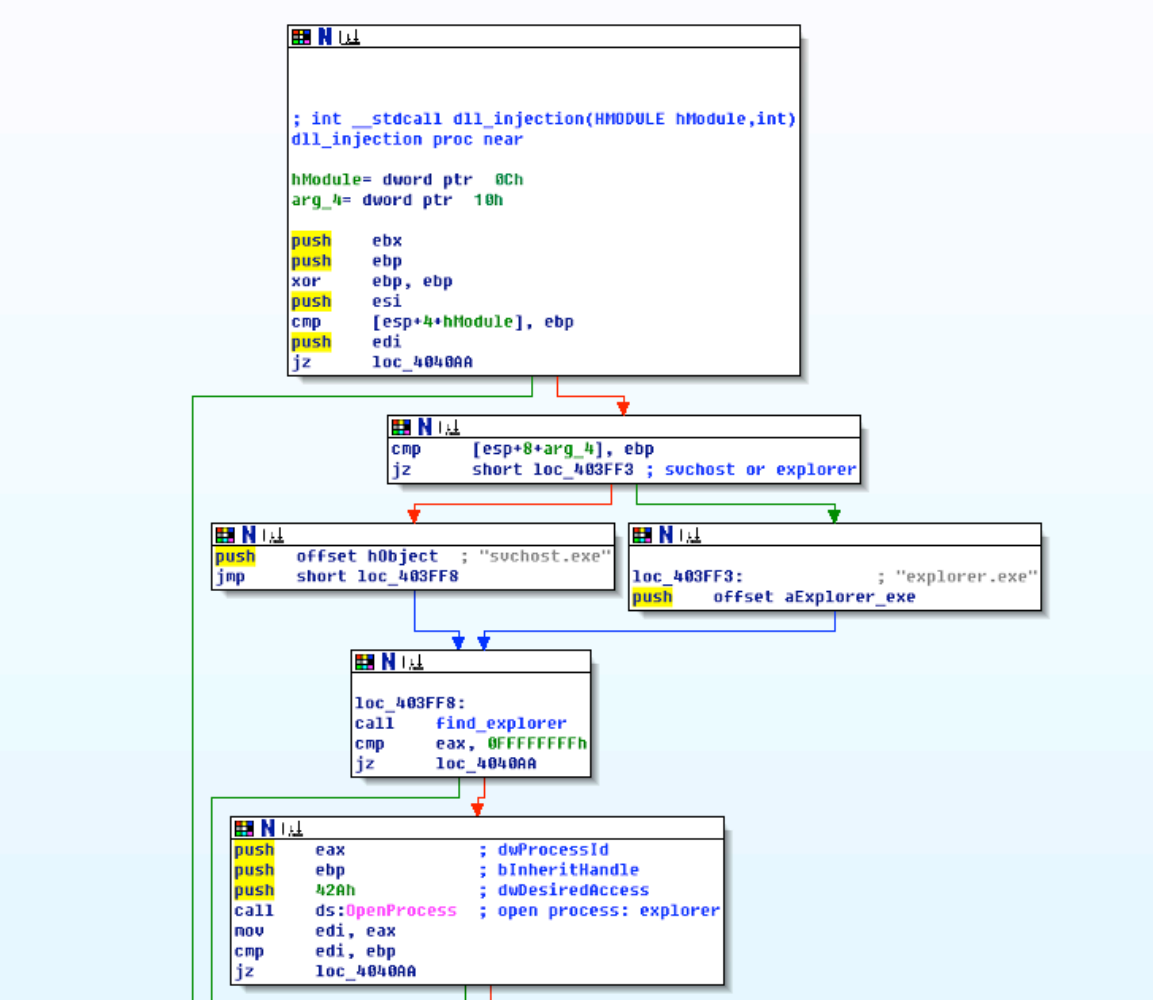


Figure 15. Instructions of injection to "explorer.exe" under IDA Pro

To continue the debugging process, the file "dg003.exe" was patched for injecting the "msvcr.dll" into the debugging "explorer.exe". A break point was set at 0x00403FBF for monitoring the scan of each running processes' PID at 0x0012EBE4. After stopping at this break point a few times, the process name of "explorer.exe" was displayed at the ASCII pane near the dumped address at 0x0012EBE4. The PID value of the system "explorer.exe" was replaced with the PID value of the debugging "explorer.exe". As an example, the system running "explorer.exe" PID of 864 (0x360) was patched with the debugging "explorer.exe" PID of 3768 (0xEB8). Because the memory address is represented in little endian under Intel CPU, to redirect the injection to the debugging "explorer.exe", the value of "60 03" was replaced with "B8 0E". (Figure 16)

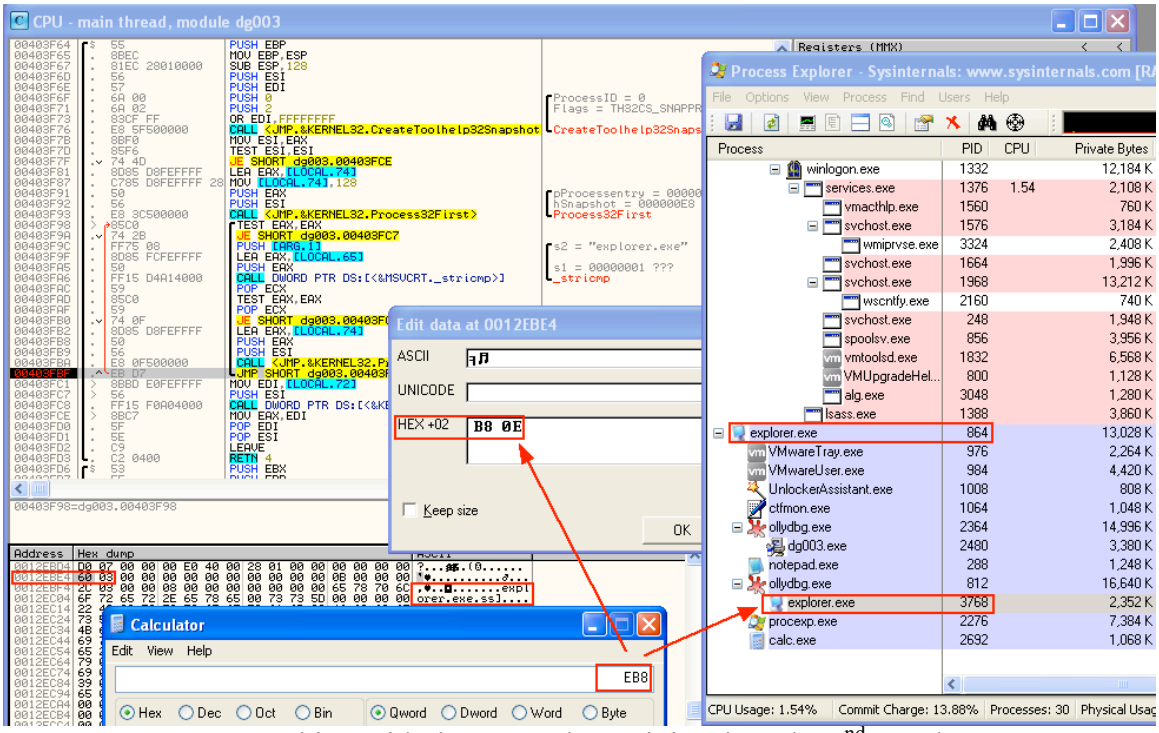


Figure 16. Patching with the PID value to inject into the 2nd "explorer.exe"

After the patching, the malware injects "msvcr.dll" to debugging "explorer.exe" by calling CreateRemoteThread API at 0x00404094. It also generates a debugger message of "注入成功! " (Literally "Injection Success!") and then terminates itself.

3.2. Analysis of The Injected "msvcr.dll"

Because the "msvcr.dll" is packed by a proprietary packing routine, it has to be manually un-packed before further code analysis. The file "msvcr.dll" is loaded into OllyDbg and a hardware break point is set at 4 bytes about the ESP (Figure 17).

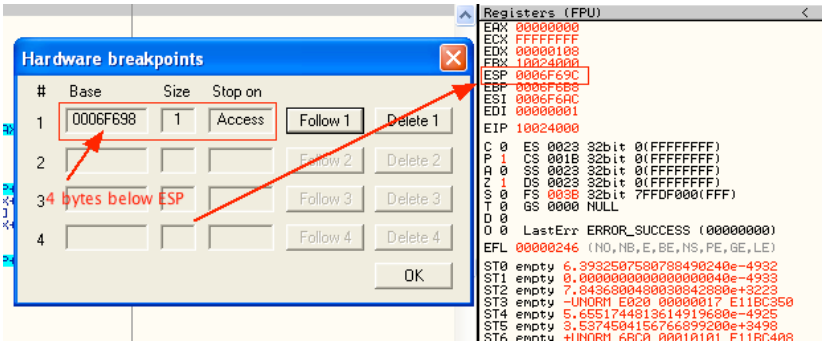


Figure 17. Setting Hardware Break Point at ESP-4

After pressing F9 to let it run, it stops at an instruction at a long jump to the address contains in EAX. (Figure 18)

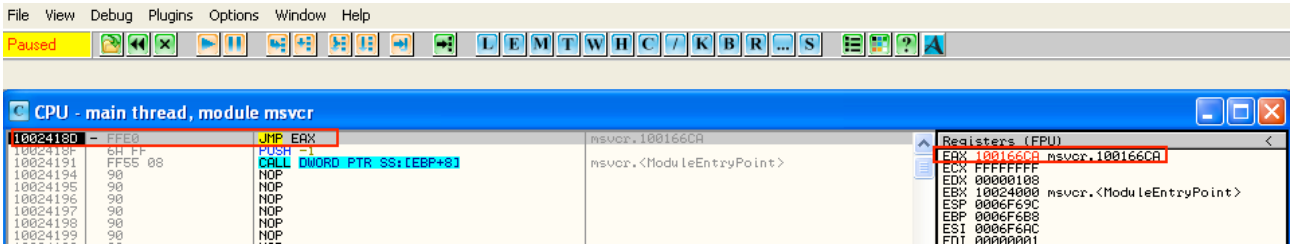


Figure 18. EAX shows the possible Original Entry Point

Pressing F7 to step into EAX at 0x100166CA, it looks like "msvcr.dll" is unpacked. The OllyDump is called to create an unpacked version of "msvcr.dll", which is loaded into IDA Pro for static code analysis.

Recalling the behavioral findings after the DLL injection (Figure 6), there are three running threads of "msvcr.dll", which are set to start at RVAs of 0xAEE0, 0xABB0 and 0xA940. After checking with the "Function View" of IDA Pro by adding base address of 0x10000000 to these RVAs, three functions at 0x10009F40, 0x1000ABB0 and 0x1000AEE0 are identified. It is clearly a good choice for setting break points at OllyDbg on the debugging "explorer.exe" (Figure 19).

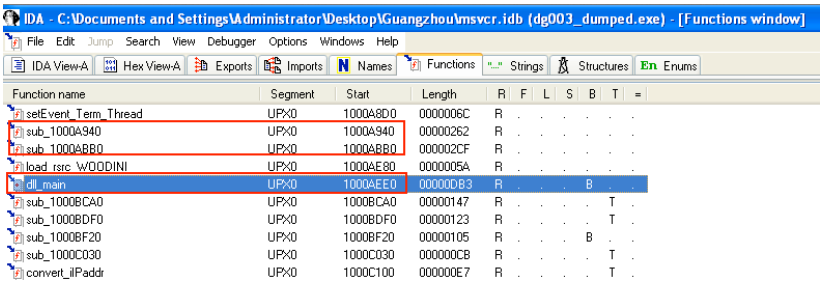


Figure 19. The three entry functions identified under IDA Pro

The following diagram shows a board overview of the order of execution and key functions of the injected "msvcr.dll" (Figure 20). A description of the performed analysis that provided these results will follow.

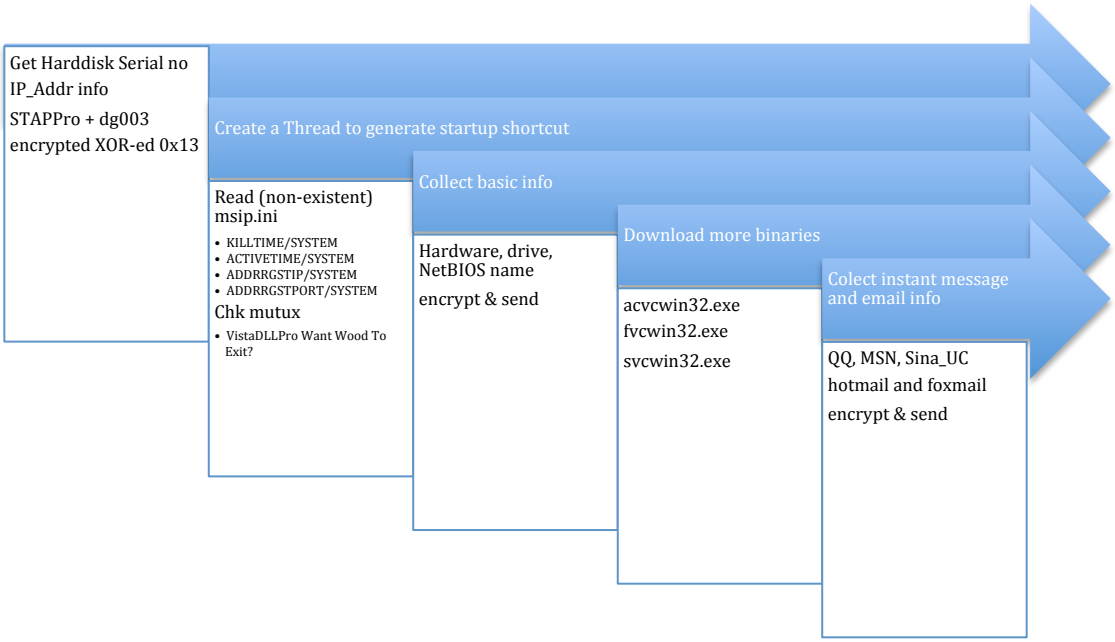


Figure 20. The key functions of the injected "msvcr.dll"

At address 0x1000AF12, the injected "msvcr.dll" creates a mutux "VistaDLLPro IS RUNNING" and generates a debugging message of "MainThread Start!" at 0x1000AF37. Then it generates a unique machine ID by referring to the network adaptor information and the hard disk volume serial number at 0x1000B013 (Figure 21).

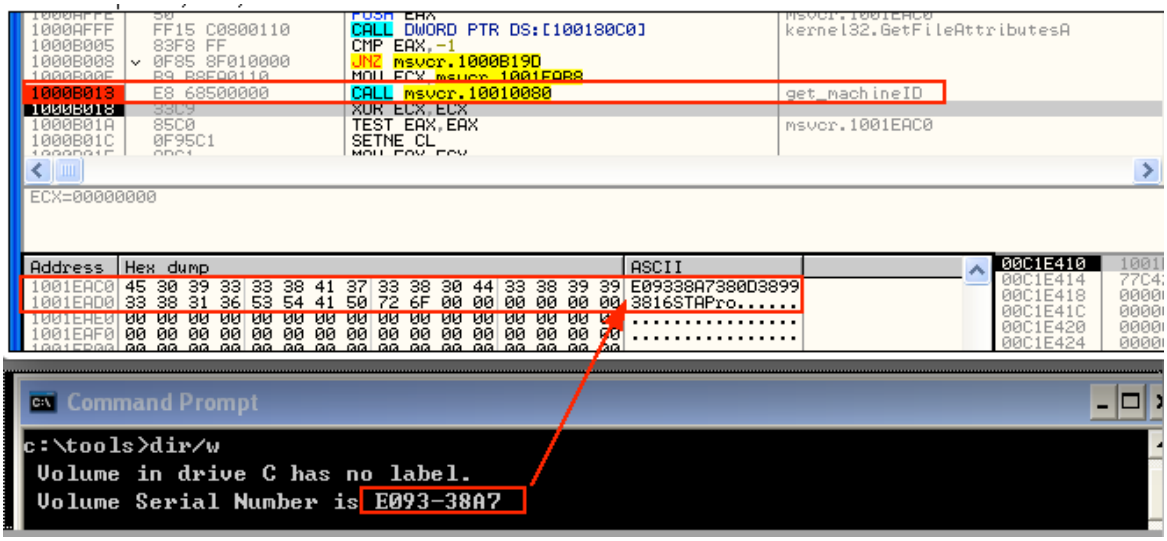


Figure 21. The hard disk volume serial number was used for generating machine ID

A strings of “STAPPro” and “dg003” is added at the end of this machine ID, it is encoded at 0x1000B0F1 and 0x1000B16D (Figure 22). A newly created file is created at C:\WINDOWS\system32\ipsecstap.dat.

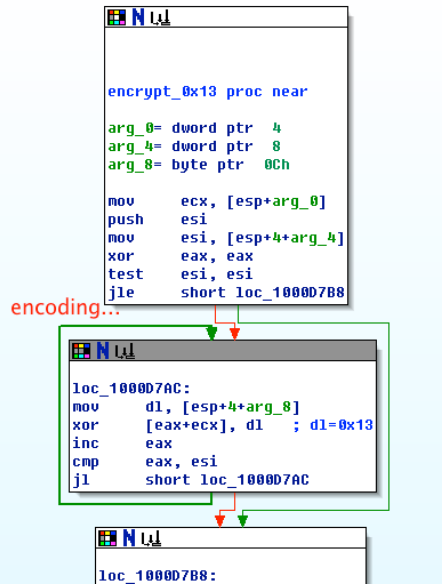


Figure 22. Encoding routine was found under IDA Pro

Then the injected “msvcr.dll” makes an API call, GetPrivateProfileString at 0x1000BD8D to retrieve strings from a non-exist file at C:\Windows\msip.ini. At 0x1000B6DA, it creates an “Assiant Thread” (The mis-spelled name which was found at 0x1000AB89 is used) to execute a function at 0x1000A940 (Figure 23). This thread will create a mutux of “VistaDLLPRo Want Wood To Exit?” and runs a loop to wait for other instructions.

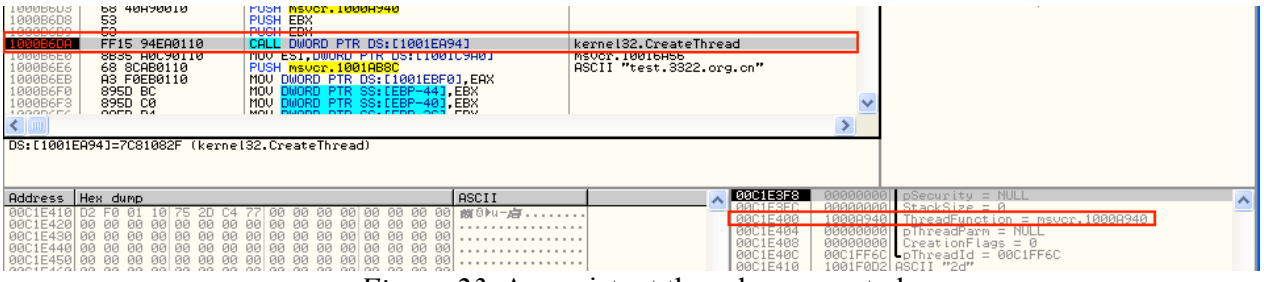


Figure 23. An assistant thread was created

The injected “msvcr.dll” tries to resolve some DNS names of test.3322.org.cn, 1.test.3322.org.cn, 2.test.3322.org.cn, 3.test.3322.org.cn and 4.test.3322.org.cn. Then,

tries to connect to a non-routable IP address of 172.16.0.61. It is believed that these are un-removed junk codes that have been previously used for program testing.

Because the injected "msvcr.dll" thread is analyzed inside a controlled environment, it runs into a loop and waits for further instructions from the C&C.

To trigger additional response, the honeyd and farpd services on the Remnux responsive box are turned on to handle the network request. Under the same behavioral studies, the injected "msvcr.dll" starts connecting to the IP address of 115.x.x.249 with TCP port number 8080. If the socket is created, it sends out some encrypted network traffic (Figure 24).

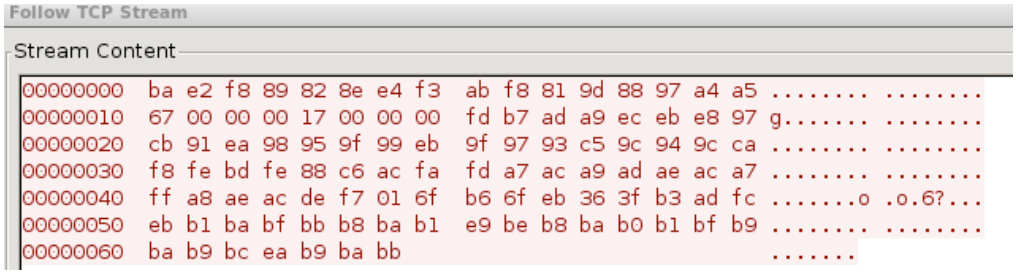


Figure 24. Encrypted traffic captured by Remnux responsive box

3.3. Behavioral and Code Analysis with Internet Access

In order to gather more responses from the actual C&C, Internet access was enabled and another session of behavioral analysis was performed.

After connected with the C&C, the injected "msvcr.dll" jumps into the function at 0x1000BB10 to 0x10001E9A. It calls various APIs including: GetVersionaExA, GetComputerNameA, GetUserNameA, GetLogicalDrives, GetDriveTypeA, GetDiskFreeSpaeExA and GetACP to collect some hard disk information from the victim's system. Before calling the encryption function, the information is temporary kept in memory (Figure 25). It also enumerates registry key of "System\CurrentControlSet\Control\ProductOption" and "Hardware\Descriptions\System\CentralProcessor\0" to gather more machine information from the infected machine.

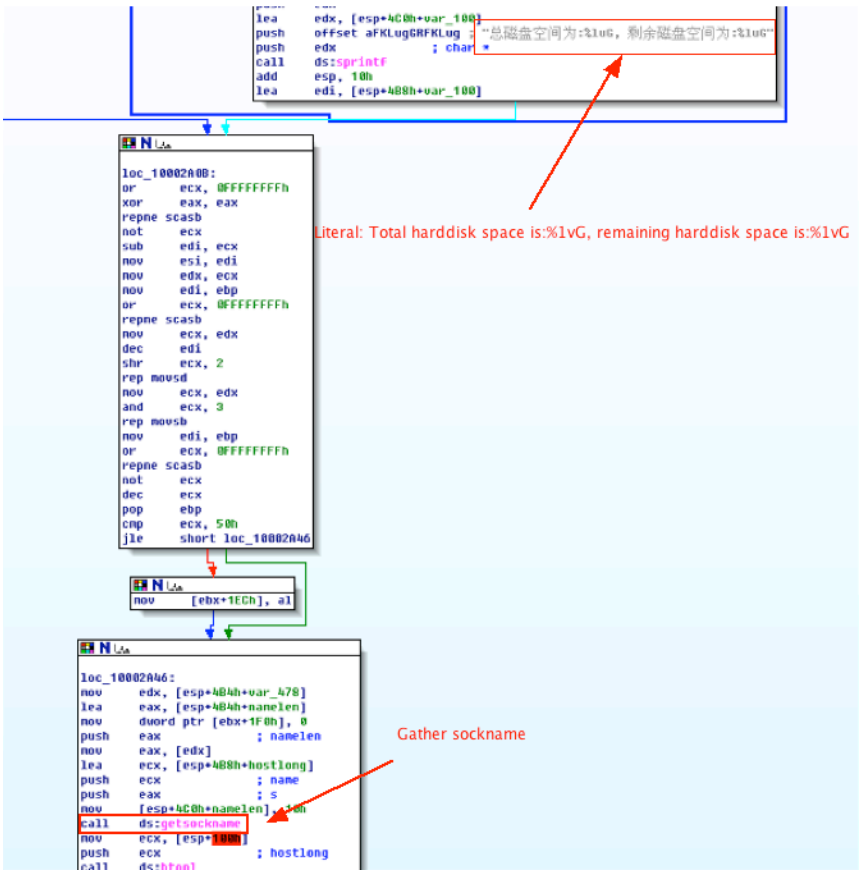


Figure 25. Gathering hard disk information under IDA Pro

The information is encrypted at 0x1000FB10. The encryption algorithm block is identified at 0x1000FB2A to 0x1000FB54.

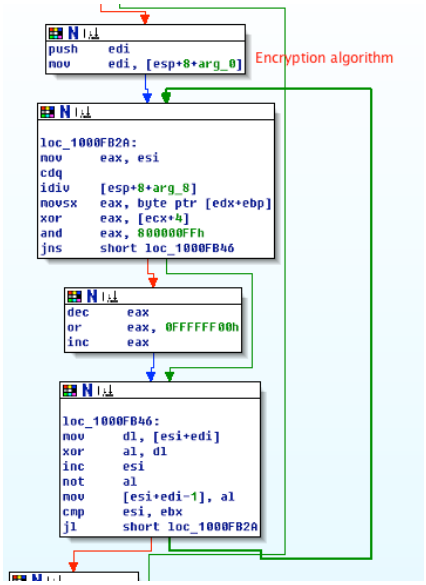


Figure 26. The encryption algorithm was found under IDA Pro

Based on this encryption algorithm, a script (Appendix II) is written for decryption part of the communication (Figure 27). It is found that the injected "msvcr.dll" first sends standard HTTP request together with the machine ID and receives a standard HTTP response from the connected C&C. Then it sends all collected information through encryption HTTP traffic to the C&C.

```
02 01 01 00 A5 4E 00 00 08 00 00 00 38 43 30 46  .□.....8C0F
39 30 31 39 33 38 30 44 33 38 39 39 33 38 31 36  9019380D38993816
53 54 41 50 72 6F 00 00 00 00 00 00 00 00 00 00  STAPro..........
00 00 00 00 32 30 30 39 30 36 30 35 30 39 32 34  ....200906050924
30 37 00 00 00 00 00 00 00 00 00 00 00 00 00 00  07..............
00 00 00 00 00 00 00 00 00 00 00 00 68 6B 30 36  ............hk06
30 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00  03..............
00 52 41 4E 32 2D 30 41 35 33 41 39 36 00 00 00  .RAN2-0A53A96...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 41 64 6D 69 6E 69 73 74 72 61 74 6F 72 00 00  .Administrator..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 4D 69 63 72 6F 73 6F 66 74 20 58 50 20 50 72  .Microsoft XPPr
6F 66 65 73 73 69 6F 6E 61 6C 2C 20 53 65 72 76  ofessional,Serv
69 63 65 20 50 61 63 6B 20 32 20 28 42 75 69 6C  ice Pack 2(Buil
64 20 32 36 30 30 29 0A 00 00 00 00 00 00 00 00  d2600).........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00..................
00 47 65 6E 75 69 6E 65 49 6E 74 65 6C 20 78 38  .GenuineIntelx8
36 20 46 61 6D 69 6C 79 20 36 20 4D 6F 64 65 6C  6 Family 6Model
20 32 33 20 53 74 65 70 70 69 6E 67 20 31 30 20   23 Stepping10
31 38 35 38 4D 48 5A 00 00 00 00 00 00 00 00 00  1858MHZ.........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 A8 03 00 00 00 00 00 00 00 00 00 00 00 00 00  .?..............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 D7 DC B4 C5 C5 CC BF D5 BC E4 CE AA 3A 33 47  .总磁盘空间为:3G
2C 20 CA A3 D3 E0 B4 C5 C5 CC BF D5 BC E4 CE AA  , 剩余磁盘空间为
3A 30 47 00 00 00 00 00 00 00 00 00 00 00 00 00  :0G..............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 27. Example of decrypted communication

At the same moment, another session of behavioral analysis is performed and the changes on the file system and registry changes are summarized below (Table 3).

Tool	Process	Location	Event	Name
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\	File added	fvcwin32.exe
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\	File added	acvcwin32.exe
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\	File added	avcwin32.exe
CaptureBAT	fvcwin32.exe	C:\WINDOWS\Debug\Data	File	drive
CaptureBAT	acvcwin32.exe	C:\WINDOWS\Debug\Data	File added	20110704145735.bmp
CaptureBAT	avcwin32.exe	C:\Documents and Settings\<user>\Application Data	File added	自动表单 .txt1309762661 (Lit. "AutoList.txt")
CaptureBAT	avcwin32.exe	C:\WINDOWS\Debug\Data	File added	SAM.dll
CaptureBAT	avcwin32.exe	C:\WINDOWS\Debug\Data	File added	system.dll
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\Data	File added	drive
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\Data	File added	Iestorage.dll
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\Data	File added	SAM.dll
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\Data	File added	system.dll
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\Data	File added	iestorage.dll
CaptureBAT	explorer.exe	C:\WINDOWS\Debug\Data	File added	drive.cab

Table 3. Further changes on the file system and registry

Applying the filtering function on Process Monitor, three new processes can be identified. (Figure 28)

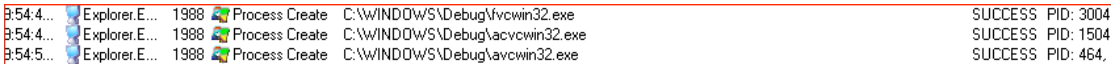


Figure 28. The debugging "Explorer.exe" creates these processes

Three binaries (fvcwin32.exe, acvcwin32.exe and avcwin32.exe) were downloaded and called by the injected "msvcr.dll". During the behavioral analysis, some files (drive.cab, iestorge.dll, SAM.dll, system.dll, 20110704145735.bmp) were created at C:\WINDOWS\Debug\Data folder. Furthermore, all files with extension of *.dll, *.v2 were uploaded to the C&C through the encrypted HTTP traffic. These files were removed by the injected "msvcr.dll" immediate after upload. In order to test the selective ability of the upload function, some files named as "Secret.v1", "Secret.v2" and "Secret.dll" were created under C:\WINDOWS\Debug\Data folder. It was found that only the file "Secret.v2" and "Secret.dll" were removed and uploaded to the C&C (Figure 29).

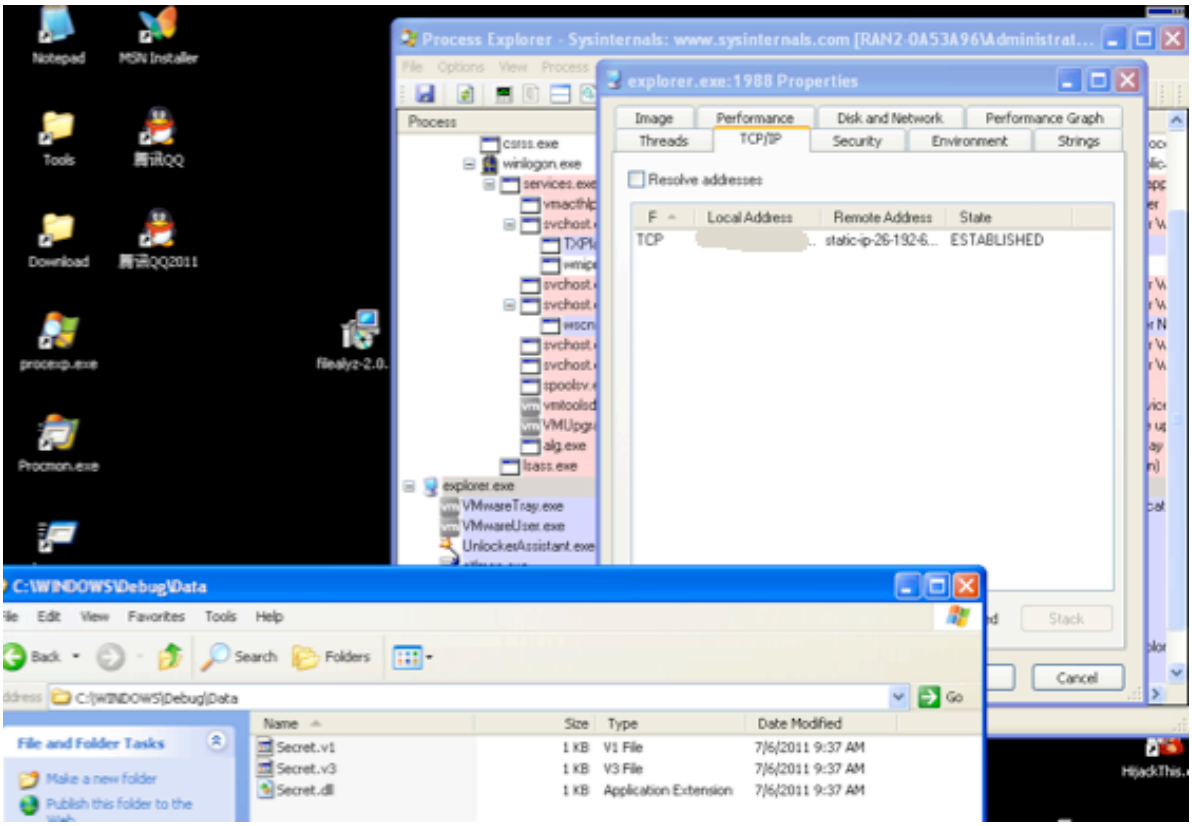


Figure 29. Only files named with extension of *.dll and *.v2 were uploaded

3.4. Trojan-Spy for Collection of Email Passwords

Before entering the main routine, the binary "avcwin32.exe" tries to create two mutux of "my lovely wood" and "SPI64 RUNNING" (Figure 30).

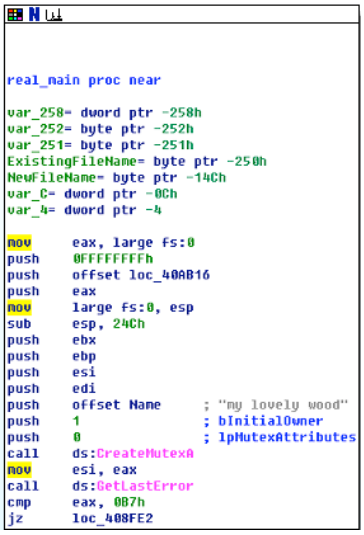


Figure 30. Mutux named "my lovely wood" was found under IDA Pro

It extracts information from the SAM file and generates a temporary file with prefix of "SAM" at C:\Documents and Settings\<user>\Application Data. It collects all passwords from "Foxmail", "Outlook express", "Outlook", "Protected Storage", "IE Form Storage", "MSN", "Passport DotNet" and from the system (Figure 31).

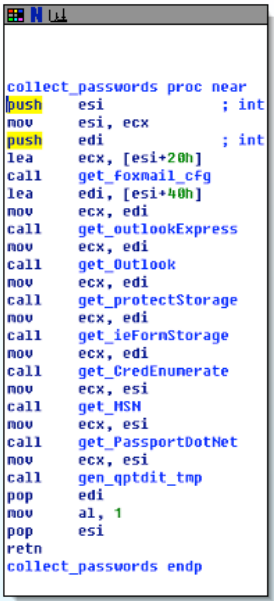


Figure 31. Malicious functions are identified under IDA Pro

All collected passwords are written to temporary files with prefix of "自动表单.txt" (Lit. "AutoList.txt"). These files are subsequently compressed in cab format and renamed as "iestorage.dll" to C:\Windows\Debug\Data folder (Figure 32).

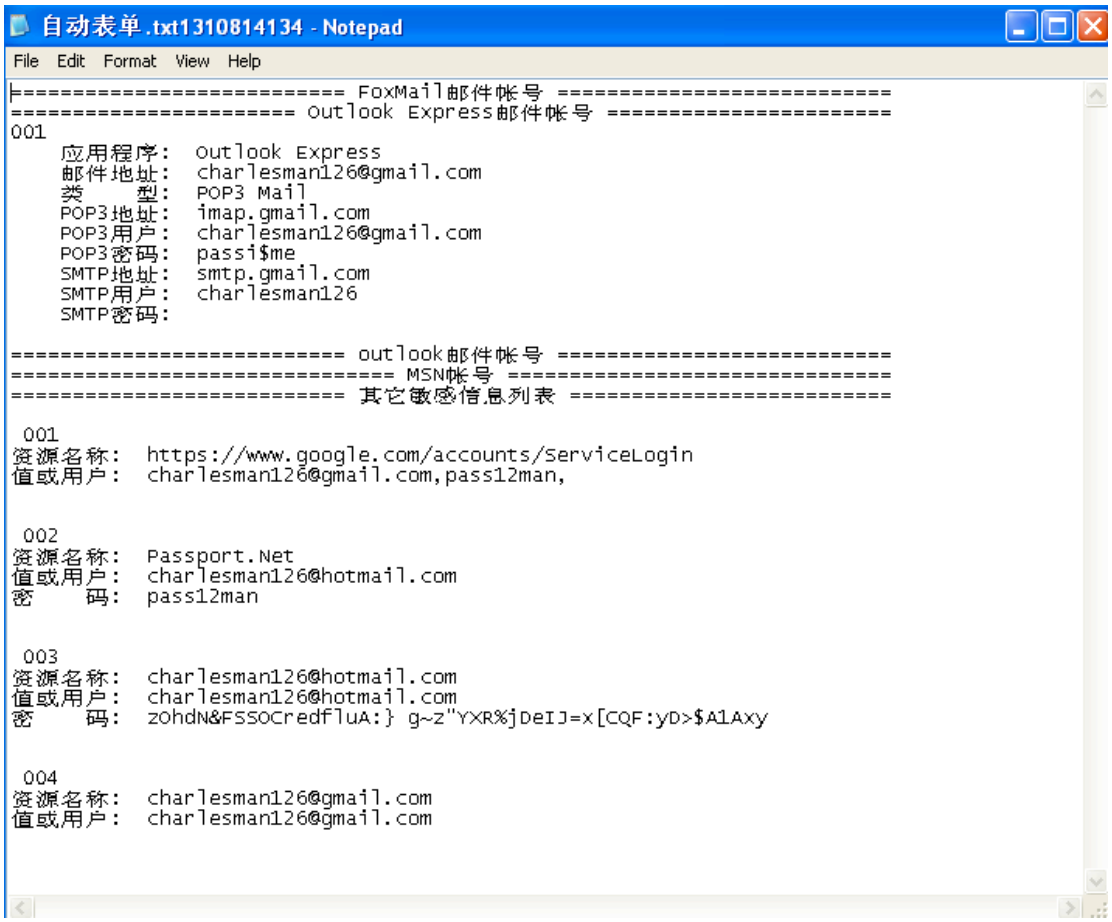


Figure 32. Contents of iestoreage.dll

Before termination, the process renamed "avcwin32.exe" as "svcwin32.exe".

3.5. Trojan-Spy for Collection of File System Details

Similar to "avcwin32.exe", the "fvcwin32.exe" process tries to create the same mutux of "my lovely wood" and "SPI64 RUNNING".

It scans all hard disk, CDROW and floppy diskette to collect all directories and file names and the respective MAC time information from these storage devices (Figure 33).

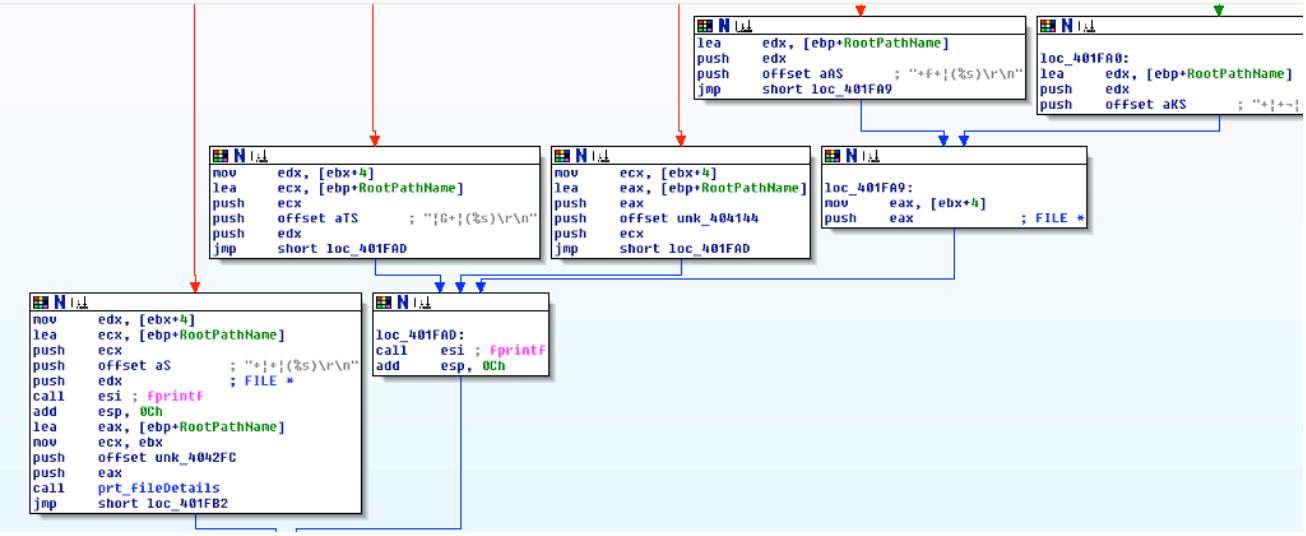


Figure 33. Directory and file names were gathered

All collected information is kept inside a file called “drive”, which is then compressed into cab format and put under C:\WINDOWS\Debug\Data. The injected “msvcr.dll” remove the un-compressed file after the upload.

3.6. Trojan-Spy for Capturing Screens

The “acvcwin32.exe” process is responsible for creation of screen captures in bitmap format for every 1000 milliseconds. All these screen captures are compressed and renamed with extension of *.v2 and put under C:\WINDOWS\Debug\Data (Figure 34).

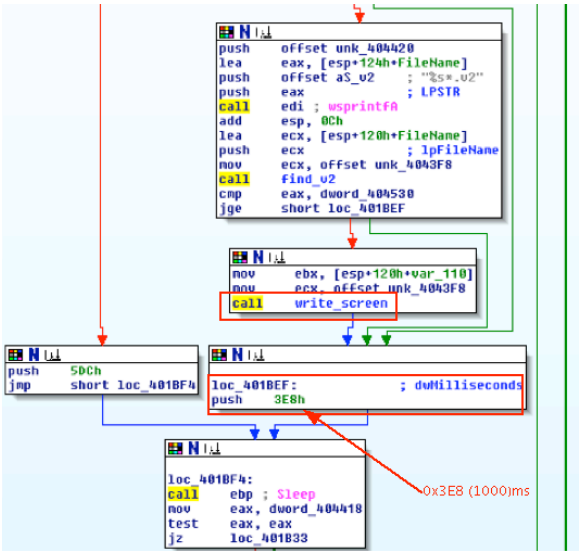


Figure 34. Bitmap is created for every 1000 milliseconds

4. Conclusion

The malware is multi-partite in nature, which includes, a dropper ("dg003.exe"), a droppee ("msvcr.dll") and at least three Trojan-Spies ("fvcwin32.exe", "acvcwin32.exe" and "avcwin32.exe"). The malware uses large amount of Windows API calls to reduce its size.

The malware uses encrypted HTTP traffic to transmit collected intelligence back to C&C, which makes it difficult to be discovered.

Indicated by the well-crafted email contents, the intruder has performed intensive prior reconnaissance on the targeted victim or has well studied the culture of the attacking organization before sending the spear-phishing email. The email contains materials of an updated political event and it was sent promptly response to the incidents.

Based on the result of the analysis, the key functions of the malware are non-profit driven (aka not motivated by financial gains), but places high emphasis on spying functions, including: generating screen captures, gathering email and messaging passwords and every directories and file names information from the victim's machine.

It bears the similar attributes as mentioned in the GhostNet report (TheSecDevGroup, 2009, p. 18-22, 24-26) and in the Operation Shady RAT (Alperovitch, 2011, p. 2-3) which indicates that it is an APT-type malware.

5. References

Alperovitch, D. (2011, Aug 2). Revealed: Operation Shady RAT. (2011). Retrieved from http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf

Brand, M., Valli, C. & Woodward, A., (2010). Malware Forensics: Discovery of the Intend of Deception. Edith Cowan University, Australian Digital Forensics Conference.

Bejtlich, R. (2011). Taosecurity Blog. Retrieved from http://taosecurity.blogspot.com/search?q=APT

Buecher, M., TiANWEi & XhmikosR (2007). Regshot (Version 1.8.2) [computing software]. Available from http://sourceforge.net/projects/regshot/

Chen Yi-Tian (陈一天) (2005, Apr 27). CHM 电子书木马制作攻略, 电脑报 .(literally "How to create Trojan horse using CHM and EXE, Computer News"). Retrieved from http://www.yesky.com/84/1942084.shtml

Cloppert, M. (2009, Jul 22). Security Intelligence: Introduction (pt 1), Computer Forensics Blog. Retrieved from http://computer-forensics.sans.org/blog/2009/07/22/security-intelligence-introduction-pt-1

Cloppert, M. (2010, Jun 21). Security Intelligence: Defining APT Campaigns, Computer Forensics Blog. Retrieved from http://computer-forensics.sans.org/blog/2010/06/21/security-intelligence-knowing-enemy/

Collake, J. (2006, Jan 1). Windows File Protection. Retrieved from http://www.bitsum.com/aboutwfp.asp

Founstone. (2000). BinText (Version 3.0) [computing software]. Available from http://www.mcafee.com/us/downloads/free-tools/bintext.aspx

Frye, S. (2011, Aug 18). Kaspersky disputes McAfee's Shady Rat Report. *TechRepublic.* Retrieved from http://www.techrepublic.com/blog/security/kaspersky-disputes-mcafees-shady-rat-report/6315

Gheorghe, C. (2008). Stud_PE (Version 2.4.0.1) [computing software]. Available from http://www.cgsoftlabs.ro/

Hex-Rays SA. (2010). IDA Pro (Freeware Version 5.0) [computing software]. Available from http://www.hex-rays.com/idapro/idadownfreeware.htm

Hoglund, G. (2011, Aug 15). Shady RAT is a Serious Business. Retrieved from http://fasthorizon.blogspot.com/2011/08/shady-rat-is-serious-business.html

Hui, A. (2011, Sep 16). (personal communication using web name as Avatar). [1st comments]. Retrieved from http://espionageware.blogspot.com/2011/09/attack-roadmap-of-apt-type-malware.html#comments

Lee, R. (2011, Aug 21). McAfee fires back at Shady RAT criticism. *SC Magazine*. (personal communication) Retrieved from http://www.scmagazineus.com/mcafee-fires-back-at-shady-rat-criticism/article/210116/

Li. F., (2011, Aug 31) Behavioral Analysis with CaptureBAT. Retrieved from http://espionageware.blogspot.com/2011/08/behavioral-analysis-with-capturebat.html

Ligh, M., Adair, S., Hartstein, B., & Richard, M. (2011a). Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code. Wiley Publishing, Inc.

Ligh, M., Adair, S., Hartstein, B., & Richard, M. (2011b). Handlediff (Version 0.2) [computing software]. Available from http://www.malwarecookbook.com/

Mandiant. (2010, Jan 25). M-Trends, the Advanced Persistent Threat. Retrieved from http://www.mandiant.com/products/services/m-trends

McAfee (2009). FileInsight (Version 2.1) [computing software]. Available from http://www.mcafee.com/us/downloads/free-tools/fileinsight.aspx

Oberhumer, M. (2008). Ultimate Packer for eXecutables (Version 3.03w) [computing software]. Available from http://upx.sourceforge.net/

Russinovich, M. (2011a). Autoruns (Version 10.06) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb963902

Russinovich, M. (2011b). Process Explorer (Version 2.93) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb896653

Russinovich, M. (2011c). Process Monitor (Version 15.0) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb896645

Russinovich, M. (2011d). ListDLLs (Version 3.1) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb896656

Russinovich, M. (2011e). TCPView (Version 3.02) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb897437

Russinovich, M. (2011f). VMmap (Version 3.1) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/dd535533

Russinovich, M. (2011g). Winobj (Version 2.21) [computing software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb896657

Snaker, Qwerton & Jibz (2008) PEiD (Version 0.95) [computing software]. Available from http://www.peid.info/

Szor, P. (2005). The Art of Computer Virus Research and Defense, Addison Wesley Professional

TheSecDevGroup. (2009, Mar 29). Tracking GhostNet: Investigating a Cyber Espionage Network. Retrieved from http://www.scribd.com/doc/13731776/Tracking-GhostNet-Investigating-a-Cyber-Espionage-Network

Valli, C. & Brand, M. (2008). The Malware Analysis Body of Knowledge (MABOK), Edith Cowan University, School of Computer and Information Science

Wireshark.org. (2010). Wireshark Network Protocol Analyzer (Version 1.2.2) [computing software]. Available from http://www.wireshark.org/

Yuschuk, O. (2004). OllyDbg (Version 1.10) [computing software]. Available from http://www.ollydbg.de/download.htm

Zealand Honeynet Project (2007). Capture BAT (Version 2.0.0) [computing software]. Available from http://www.honeynet.org/node/315

Zeltser, L. (2007). Reverse Engineering Malware: Tools and Techniques Hands –On. Bethesda: SANS Institute.

Zeltser, L. (2011). REMnux: A Linux Distribution for Reverse-Engineering Malware (Version 2.0) [computing software]. Available from http://zeltser.com/remnux/

Zimmer, D. (2001). Malcode Analysis Pack (Version 1.0) [computing software]. Available from http://www.woodmann.com/collaborative/tools/index.php/Malcode_Analysis_Pack

Appendix I

On the Remnux responsive box, the honeyd configuration file was found at /etc/honeypot/honeyd.conf (Figure 35) and the configuration file of farpd was found at /etc/default/farpd (Figure 36).

honeyd.conf

/etc/honeypot/honeyd.conf

```
create default
#set default default tcp action block
set default default udp action block
#set default default icmp action block

add default tcp port 80 "sh /usr/share/honeyd/scripts/web.sh"
add default tcp port 6666 proxy 127.0.0.1:6666
add default tcp port 3127 proxy 127.0.0.1:3127
add default tcp port 8080 proxy 127.0.0.1:8080
```

Figure 35. Configuration of honeyd

farpd

/etc/default/farpd

```
# Options for the Fake arp daemon
#
# Interface where farpd will listen in
INTERFACE="eth0"

# Network for which fake arp daemon will answer to
# You can leave this empty but then the daemon will try to take over
# all the network addresses in that interface
#
# Note: Default 'unconfigured' means that farpd will not startup, you
# have to setup a valid value here (can be empty)
NETWORK=""

# Time to wait for the daemon to die before restarting it
# (in seconds)
# DODTIME=5
```

Figure 36. Configuration of farpd

Appendix II

```
import os,sys,glob,pcap,dpkt
from dpkt import ip,tcp
from socket import inet_ntoa
import binascii

def preparekey ():
   orikey = "c9273029a6028971214b123b54b0d72d"
   seckey = 0x61
   newkey = []
   snewkey = []
   for ac in orikey:
       atmp = ord(ac)^seckey
       newkey.append(atmp)
       snewkey.append(hex(atmp))
   return newkey

def decode (data,key):
   lenkey = len(key)
   lendata = len(data)
   newdata = []
   snewdata = []
   hexnewdata = []
   j = -1
   for i in range(0,lendata):
       ab = data[i]
       j = j + 1
       if i>15 and i<24:
          newdata.append(ord(ab))
          hexnewdata.append(hex(ord(ab)))
          snewdata.append(ab)
          continue
       elif i==24:
           j = 0
       rab = ~ord(ab)
       if rab < 0:
          rab = rab + 256
       ikey = j % lenkey
       ak = key[ikey]
       av = ak^rab
       newdata.append(av)
       hexnewdata.append(hex(av))
       snewdata.append(chr(av))
   alldata = []
   for i in range(0,lendata):
       alldata.append([newdata[i],hexnewdata[i],snewdata[i]])
   return newdata,hexnewdata,snewdata,alldata

def main(adir):
        files = glob.glob(adir+'*.pcap')
        CCSrvIP = "115.x.x.249"


        key = preparekey()
        for afile in files:
            print afile
            outbinfilename = afile.replace('.pcap','_out.bin')
            outbinf = open(outbinfilename,'wb')
            outhexfilename = afile.replace('.pcap','_out.hex')
            outhexf = file(outhexfilename,'w')
            outdecodefilename = afile.replace('.pcap','_out.decode')
            outdecodef = file(outdecodefilename,'w')
            outdecodetxtfilename = afile.replace('.pcap','_out.decode_txt')
            outdecodetxtf = file(outdecodetxtfilename,'w')
            indecodefilename = afile.replace('.pcap','_in.decode')
            indecodef = file(indecodefilename,'w')
            indecodetxtfilename = afile.replace('.pcap','_in.decode_txt')
            indecodetxtf = file(indecodetxtfilename,'w')
            inbinfilename = afile.replace('.pcap','_in.bin')
            inbinf = open(inbinfilename,'wb')
            inhexfilename = afile.replace('.pcap','_in.hex')
            inhexf = file(inhexfilename,'w')

            apcap = pcap.pcap(afile)
            apcap.setfilter('tcp')
            for ts,buf in apcap:
               try:
                   eth = dpkt.ethernet.Ethernet(buf)
                   ip = eth.data
                   srcip = str(inet_ntoa(ip.src)).strip()
                   tcp = ip.data
                   tcplen = str(ip.len - (ip.v_hl & 0xf)*4 - (tcp.off_x2 >> 4)*4)
                   if int(tcplen) == 0:
                      continue
                   tcpdata = tcp.data
                   if srcip == CCSrvIP:
                      newdata,hexnewdata,snewdata,alldata = decode (tcpdata,key)
                      inbinf.write(tcpdata)
                      print >> inhexf, binascii.hexlify(tcpdata)
                      print >> indecodef, str(alldata)
                      print >> indecodetxtf,snewdata
                   else:
                      newdata,hexnewdata,snewdata,alldata = decode (tcpdata,key)
                      outbinf.write(tcpdata)
                      print >> outhexf, binascii.hexlify(tcpdata)
                      print >> outdecodef, str(alldata)
                      print >> outdecodetxtf,snewdata
               except dpkt.UnpackError as aerror:
                   print aerror
            outbinf.close()
            del outhexf
            inbinf.close()
            del inhexf
            del indecodef

            del outdecodetxtf
            del indecodetxtf

if __name__ == '__main__':
     topdir = sys.argv[1].strip()
     if topdir[-1] != '/':
        topdir = topdir + '/'
     main(topdir)
```

Last Updated: April 8th, 2014

Upcoming Training



SANS Security West 2014	San Diego, CA	May 08, 2014 - May 17, 2014	Live Event
Mentor Session - FOR 610	Columbia, MD	May 21, 2014 - Jul 23, 2014	Mentor
Digital Forensics & Incident Response Summit	Austin, TX	Jun 03, 2014 - Jun 10, 2014	Live Event
Community SANS Ottawa	Ottawa, ON	Jun 16, 2014 - Jun 21, 2014	Community SANS
SANSFIRE 2014	Baltimore, MD	Jun 21, 2014 - Jun 30, 2014	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201407,	Jul 14, 2014 - Aug 20, 2014	vLive
SANS Virginia Beach 2014	Virginia Beach, VA	Aug 18, 2014 - Aug 29, 2014	Live Event
SANS Baltimore 2014	Baltimore, MD	Sep 22, 2014 - Sep 27, 2014	Live Event
SANS DFIR Prague 2014	Prague, Czech Republic	Sep 29, 2014 - Oct 11, 2014	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201410,	Oct 13, 2014 - Nov 19, 2014	vLive
Community SANS Paris @ HSC - FOR610 (in French)	Paris, France	Nov 24, 2014 - Nov 28, 2014	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced