

# Rerverse Engineer: Lab 5 Report

Jesus Lopez, Alejandro Rodriguez, Eduardo Menendez  
Dr. Saidur Rahman

November 3, 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Volatility</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Installation . . . . .	3
2.2.1	Usage . . . . .	4
2.3	Exercise 1: Identify Profile and List Processes . . . . .	4
2.3.1	Objective . . . . .	4
2.3.2	Taks . . . . .	4
2.3.3	Results . . . . .	5
2.4	Exercise 2: Detect Hidden Processes with psscan and psxview . . . . .	6
2.4.1	Objective . . . . .	6
2.4.2	Taks . . . . .	6
2.4.3	Results . . . . .	6
2.5	Exercise 3 . . . . .	8
2.5.1	Objective . . . . .	8
2.5.2	Taks . . . . .	8
2.5.3	Results . . . . .	8
<b>3</b>	<b>Dshell</b>	<b>10</b>
3.1	Introduction . . . . .	10
3.2	Installation . . . . .	10
3.2.1	Usage . . . . .	11
3.3	Disclaimer . . . . .	11
3.4	Exercise 1: Analyze HTTP Traffic . . . . .	12
3.4.1	Objective . . . . .	12
3.4.2	Task . . . . .	12
3.4.3	Results . . . . .	12
3.5	Exercise 2: Detect DNS Tunneling . . . . .	13
3.5.1	Objective . . . . .	13
3.5.2	Task . . . . .	13
3.5.3	Results . . . . .	14

# Chapter 1

## Introduction

In this lab we are going to be exploring the tools of **Volatility** and **Dshell**. These are great cybersecurity tools that are used in the real world and also in some CTF's. For the purpose of doing this lab, we used Kali linux, the system specification is the following:

A terminal window on a Kali Linux system. The user has run the 'screenfetch' command, which displays a ASCII art logo of Kali Linux on the left and system information on the right. The system information includes OS, Kernel, Uptime, Packages, Shell, Resolution, DE, WM, WM Theme, GTK Theme, Icon Theme, Font, Disk, CPU, GPU, and RAM.

```
(sagar@kali)-[~]
$ screenfetch
.....
..,:ccc,.
.....';lx0.
.....';ld;
.....';:;:;,,x,
..'''.
0Xxoc:,. ...
,ONkc;,,cok0dc',.
. ....
OMo      ': o.
dMc      :00;
0M.      .:o.
;Wd
;XO,
,d00dlc;,.
..';,:cd00d::,.
.:d;.:':;.
'd, .'
;l ..
.o
c
'
.

sagar@kali
OS: Kali Linux
Kernel: x86_64 Linux 6.8.11-amd64
Uptime: 2h 42m
Packages: 2789
Shell: zsh 5.9
Resolution: 2558x1280
DE: Xfce
WM: Xfwm4
WM Theme: Kali-Dark
GTK Theme: Kali-Dark [GTK2]
Icon Theme: Flat-Remix-Blue-Dark
Font: Cantarell 11
Disk: 19G / 51G (38%)
CPU: 11th Gen Intel Core i7-11700KF @ 4x 3.6GHz
GPU: VMware SVGA II Adapter
RAM: 2785MiB / 5753MiB

(sagar@kali)-[~]
$
```

The instructions on how to install each one of the tools are going to be on each one of their sections, as well a guide on how to use them.

## Chapter 2

# Volatility

### 2.1 Introduction

Volatility3 is a very powerful memory forensics tool, that is used to extract memory images of Windows, MacOS, and Linux systems. of course, it is reasonable to mention that there is a huge community writing third-party plugins for volatility3.

### 2.2 Installation

To install volatility we have to first clone the main repository from their github. This is the command to use:

```
1 git clone https://github.com/volatilityfoundation/volatility3
   .git
```

From here there are two options:

1. You can extract it and then let it distribute everything in your system, so that you can use the tool anywhere you like.
2. Call vol.py from the folder that was extracted everytime you like to use the tool.

The main difference is that the first one is quite difficult to remove all the files and clean it, where the second one you only need to erase the folder to erase the tool from your system. For installation purposes, if having trouble you can refer to this page [INSTALLATION PAGE](#)

## 2.2.1 Usage

Here are the options of volatility3:

```
An open-source memory forensics framework
options:
-h, --help            Show this help message and exit, for specific plugin options use 'volatility <pluginname> --help'
-c CONFIG, --config CONFIG
                        Load the configuration from a json file
--parallelism [{processes,threads,off}]
                        Enables parallelism (defaults to off if no argument given)
-e EXTEND, --extend EXTEND
                        Extend the configuration with a new (or changed) setting
-p PLUGIN_DIRS, --plugin-dirs PLUGIN_DIRS
                        Semi-colon separated list of paths to find plugins
-s SYMBOL_DIRS, --symbol-dirs SYMBOL_DIRS
                        Semi-colon separated list of paths to find symbols
-v, --verbosity        Increase output verbosity
-l LOG, --log LOG      Log output to a file as well as the console
-o OUTPUT_DIR, --output-dir OUTPUT_DIR
                        Directory in which to output any generated files
-q, --quiet            Remove progress feedback
-r RENDERER, --renderer RENDERER
                        Determines how to render the output (quick, none, csv, pretty, json, jsonl)
-f FILE, --file FILE  Shorthand for --single-location=file:// if single-location is not defined
--write-config         Write configuration JSON file out to config.json
--save-config SAVE_CONFIG
                        Save configuration JSON file to a file
--clear-cache          Clears out all short-term cached items
--cache-path CACHE_PATH
                        Change the default path (/home/vidza/.cache/volatility3) used to store the cache
--offline             Do not search online for additional JSON files
-u URL, --remote-isf-url URL
                        Search online for ISF json files
--filters FILTERS     List of filters to apply to the output (in the form of [+]-columnname,pattern[!])
--hide-columns [HIDE_COLUMNS ...]
                        Case-insensitive space separated list of prefixes to determine which columns to hide in the output if provided
--single-location SINGLE_LOCATION
                        Specifies a base location on which to stack
--stackers [STACKERS ...]
                        List of stackers
--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]
                        Specifies a list of swap layer URIs for use with single-location

Plugins:
For plugin specific options, run 'volatility <plugin> --help'
```

If desired, you can look at an introduction of memory forensics with volatility [\[LINK\]](#)

For these exercises we are going to be using images that were provided by the instructor.

## 2.3 Exercise 1: Identify Profile and List Processes

### 2.3.1 Objective

For task, Use the imageinfo and pslist plugins to identify the OS profile and list all active processes.

### 2.3.2 Tasks

1. Run the following command to determine the correct OS profile:

```
1 volatility -f example.mem imageinfo
```

2. Use the recommended profile to list all running processes:

```
1 volatility -f example.mem --profile=Win7SP1x64 pslist
```

3. Identify the top 3 processes consuming the most resources
4. Research any unfamiliar process names and note their purpose.
5. Record the PID, PPID, and creation time of any suspicious processes.

### 2.3.3 Results

first we are going to run:

```
1 sudo vol -f c2.img windows.info.Info
```

This is the result:

```
(sagar@kali)~/Desktop/memory_analysis
$ cat c2.txt
Volatility 3 Framework 2.11.0

Variable      Value
-----
Kernel Base   0x8263b000
DTB           0x185000
Symbols file:  ///usr/local/lib/python3.11/dist-packages/volatility3-2.11.0-py3.11.egg/volatility3/symbols/windows/ntkrpmp.pdb/92D32EE7188A4CB3AB23EDA0CB0F9D7B-2.json.xz
Is64Bit       False
IsPAE         True
layer_name    0 WindowsIntelPAE
memory_layer  1 FileLayer
KdDebuggerDataBlock 0x82767c78
NTBuildLab    7601.23915.x86fre.win7sp1_ldr.17
CSDVersion    1
KdVersionBlock 0x82767c50
Major/Minor   15.7601
MachineType   332
KeNumberProcessors 1
SystemTime    2019-04-25 20:40:40+00:00
NtSystemRoot  C:\Windows
NtProductType NtProductWinNt
NtMajorVersion 6
NtMinorVersion 1
PE_MajorOperatingSystemVersion 6
PE_MinorOperatingSystemVersion 1
PE_Machine    332
PE_TimeDateStamp Wed Sep 13 14:47:57 2017

(sagar@kali)~/Desktop/memory_analysis
$
```

to check for processes we are going to use the following command:

```
1 sudo vol -f c2.img windows.psscan.PsScan
```

from there we are going to get a table with the PID, the PPID, the ImageFileName and more things:

```
(vidza@kali)~/Desktop/volatility/volatility3
$ python3 vol.py -f ./c2.img windows.psscan.PsScan
Volatility 3 Framework 2.11.0
Progress: 100.00

PID      PPID      ImageFileName      PDB      Scanning      Finished      Threads      Handles      SessionId      Wow64      CreateTime      ExitTime      File output
-----
2352     2260     SearchProtocol     0x1e63ed28  7      257      1      False      2019-04-25 20:40:10.000000 UTC      N/A      Disabled
2376     2260     SearchFilterHost  0x1e644810  5      85      0      False      2019-04-25 20:40:10.000000 UTC      N/A      Disabled
1608     320      conhost.exe       0x1e87ed28  2      33      0      False      2019-04-25 20:39:13.000000 UTC      N/A      Disabled
1632     452     wls.exe           0x1e885988  4      46      0      False      2019-04-25 20:39:13.000000 UTC      N/A      Disabled
1640     1588     shd.exe           0x1e890010  6      106      0      False      2019-04-25 20:39:13.000000 UTC      N/A      Disabled
1924     452     svchost.exe       0x1e8e6a40  5      92      0      False      2019-04-25 20:39:14.000000 UTC      N/A      Disabled
1840     452     sspsvc.exe        0x1e8ecb40  5      146      0      False      2019-04-25 20:39:14.000000 UTC      N/A      Disabled
1216     452     taskhost.exe      0x1e94cb00  10     186      1      False      2019-04-25 20:40:03.000000 UTC      N/A      Disabled
1684     828     dm.exe            0x1e977988  5      75      1      False      2019-04-25 20:40:03.000000 UTC      N/A      Disabled
1448     1592     explorer.exe      0x1e9a1b38  29     667      1      False      2019-04-25 20:40:03.000000 UTC      N/A      Disabled
2052     1448     VBoxTray.exe      0x1e9e9780  14     146      1      False      2019-04-25 20:40:04.000000 UTC      N/A      Disabled
624     452     VBoxService.exe  0x1e9d7510  12     116      0      False      2019-04-25 21:39:00.000000 UTC      N/A      Disabled
1524     452     cygrunsrv.exe     0x1e9c910  6      101      0      False      2019-04-25 20:39:12.000000 UTC      N/A      Disabled
676     452     svchost.exe       0x1eafef38  7      230      0      False      2019-04-25 20:39:10.000000 UTC      N/A      Disabled
722     452     svchost.exe       0x1eb02830  19     361      0      False      2019-04-25 20:39:10.000000 UTC      N/A      Disabled
2540     2504     JavaProgram.exe   0x1eb2c9f8  3      99      1      False      2019-04-25 20:40:33.000000 UTC      N/A      Disabled
828     452     svchost.exe       0x1eb47978  18     377      0      False      2019-04-25 20:39:10.000000 UTC      N/A      Disabled
856     452     svchost.exe       0x1eb4f830  21     339      0      False      2019-04-25 20:39:10.000000 UTC      N/A      Disabled
2260     452     SearchIndexer.exe 0x1eb00258  14     657      0      False      2019-04-25 20:40:10.000000 UTC      N/A      Disabled
884     452     svchost.exe       0x1eb536d0  32     704      0      False      2019-04-25 20:39:10.000000 UTC      N/A      Disabled
968     452     svchost.exe       0x1eb6e030  5      114      0      False      2019-04-25 20:39:11.000000 UTC      N/A      Disabled
1224     452     svchost.exe       0x1eb0bd48  16     357      0      False      2019-04-25 20:39:11.000000 UTC      N/A      Disabled
1232     452     spoolsv.exe       0x1eb036f8  15     291      0      False      2019-04-25 20:39:12.000000 UTC      N/A      Disabled
1260     452     svchost.exe       0x1ebfac30  24     337      0      False      2019-04-25 20:39:12.000000 UTC      N/A      Disabled
364     348     csrss.exe         0x1ed01850  7      176      1      False      2019-04-25 21:39:06.000000 UTC      N/A      Disabled
356     312     wininit.exe       0x1ed01d28  4      80      0      False      2019-04-25 21:39:06.000000 UTC      N/A      Disabled
392     348     winlogon.exe      0x1edd0d28  6      119      1      False      2019-04-25 21:39:07.000000 UTC      N/A      Disabled
460     356     lsass.exe         0x1edd0b30  7      536      0      False      2019-04-25 21:39:07.000000 UTC      N/A      Disabled
452     356     services.exe      0x1edd0b78  14     214      0      False      2019-04-25 21:39:07.000000 UTC      N/A      Disabled
468     356     ls.exe            0x1ed44780  11     154      0      False      2019-04-25 21:39:07.000000 UTC      N/A      Disabled
1348     452     svchost.exe       0x1ed6ea08  12     176      0      False      2019-04-25 20:39:12.000000 UTC      N/A      Disabled
1376     452     svchost.exe       0x1ed78a08  8      164      0      False      2019-04-25 20:39:12.000000 UTC      N/A      Disabled
564     452     svchost.exe       0x1eda4608  13     358      0      False      2019-04-25 21:39:07.000000 UTC      N/A      Disabled
320     312     csrss.exe         0x1f089418  9      383      0      False      2019-04-25 21:39:06.000000 UTC      N/A      Disabled
252     4     smss.exe          0x1fa9020  2      29      N/A      False      2019-04-25 21:39:05.000000 UTC      N/A      Disabled
1588     1524     cygrunsrv.exe     0x1fc74c78  0      0      0      False      2019-04-25 20:39:12.000000 UTC      2019-04-25 20:39:13.000000 UTC      Disabled
4      0      System            0x1ffefb60  65     486      N/A      False      2019-04-25 21:39:05.000000 UTC      N/A      Disabled
```

Now, to find the processes that consumes the most, here is a ranking based on handle counts:

1. svchost.exe (PID 884) - 704 Handles
2. explorer.exe (PID 1448) - 667 Handles
3. SearchIndexer.exe (PID 2260) - 657 Handles

For the last section of this task, some weird processes can be **sshd.exe** and **JavaProgram.exe**, the sshd is a process from OpenSSH, the problem might begin if the system was not intended to run SSH services, this might be an indicative that the computer is compromised and has remote access. from the otherside, JavaProgra.exe lacks of descroption on what is doing, this might be a malware running disguised as a java appliacation.

## 2.4 Exercise 2:Detect Hidden Processes with psscan and psxview

### 2.4.1 Objective

for the exercise 2, Use psscan and psxview plugins to detect hidden or terminated processes.

### 2.4.2 Taks

1. Run psscan to detect hidden and terminated processes:

```
1 sudo vol -f c2.img windows.psscan.PsScan
```

2. Compare the output with pslist. Identify processes in psscan but not in pslist.
3. Use psxview to find discrepancies between visibility views:

```
1 sudo vol -f c2.img windows.psxview
```

4. List any processes found by psscan but not in pslist
5. Identify discrepancies in psxview output and explain potential reasons.
6. Record the PID and offset of any suspicious processes.

### 2.4.3 Results

Something to mention is that on the first exercise we used psscan, it kind of gives us the same results. in this image, we did not noticed any hidden processes, pslist wont list compared to psscan. to run psxview we are going to be using the next command:

```
1 sudo vol -f c2.img windows.psxview
```

The result of running this command is the following:

```
(vidzza@kali)-[~/Desktop/volatility/volatility3]
$ python3 vol.py -f ./c2.img psxview
Volatility 3 Framework 2.11.0
Progress: 100.00
Offset(Virtual) Name PID PDB scanning finished pslist psscan thrdscan csrss Exit Time
0x84fe9780 VBoxTray.exe 2052 True False True True True
0x1e9e9780 VBoxTray.exe 2052 False True False False False
0x1ed01d28 wininit.exe 356 False True False False False
0x84b44708 lsm.exe 468 True False True True True
0x84b6ea08 svchost.exe 1348 True False True True True
0x84b78a08 svchost.exe 1376 True False True True True
0x84e85988 wlms.exe 1632 True False True True True
0x84f77908 dwm.exe 1684 True False True True True
0x1e885988 wlms.exe 1632 False True False False False
0x1e977908 dwm.exe 1684 False True False False False
0x1ed44708 lsm.exe 468 False True False False False
0x84de36f8 spoolsv.exe 1232 True False True True True
0x84cec910 cygrunsrv.exe 1524 True False True True True
0x85044810 SearchFilterHo 2376 True False True True True
0x1e644810 SearchFilterHo 2376 False True False False False
0x1eaec910 cygrunsrv.exe 1524 False True False False False
0x1ebe36f8 spoolsv.exe 1232 False True False False False
0x84cd7518 VBoxService.ex 624 True False True True True
0x84cfe518 svchost.exe 676 True False True True True
0x84609418 csrss.exe 320 True False True False False
0x1ead7518 VBoxService.ex 624 False True False False False
0x1eafe518 svchost.exe 676 False True False False False
0x1ed0dd28 winlogon.exe 392 False True False False False
0x1f009418 csrss.exe 320 False True False False False
0x840a9020 smss.exe 252 True False True False False
0x1f6a9020 smss.exe 252 False True False False False
0x1eb47978 svchost.exe 828 False True False False False
0x84b01d28 wininit.exe 356 True False True True True
0x84b0dd28 winlogon.exe 392 True False True True True
0x84e7ed28 conhost.exe 1608 True False True True True
0x8503ed28 SearchProtocol 2352 True False True True True
0x84d2c9f8 JavaProgram.ex 2540 True False True True True
0x1e63ed28 SearchProtocol 2352 False True False False False
0x1e87ed28 conhost.exe 1608 False True False False False
0x1eb2c9f8 JavaProgram.ex 2540 True False True False False
0x84b3d030 lsass.exe 460 True False True True True
0x84d28030 svchost.exe 732 True False True True True
0x84d4f030 svchost.exe 856 True False True True True
0x84d6e030 svchost.exe 968 True False True True True
0x84b3db78 services.exe 452 True False True True True
0x84dfac30 svchost.exe 1260 True False True True True
```

Something to mention in the discrepancies between the commands psscan and psxview, is that on psxview it gives us the virtual offset of the process. where on the other command gives us PPID along with the PID. in addition to this on psscan we can get the threads and the handles where on psxview we do not. also, We as a team think that on the columns we can see which of the process can be able to be seen using pslist and some of them can be seen using psscan.

For the PID and the offset of suspicious processes, we are going to be using the ones from the first task. so for sshd.exe is the following:



```
(sagar@kali)-[~/Desktop/memory_analysis]
$ sudo vol -f c2.img windows.psxview | grep sshd
[sudo] password for sagar:
0x84e90030 100.0sshd.exe      1640    True    False   True    True
0x1e890030      sshd.exe      1640    False   True    False   False
```

and JavaProgram.exe:

```
(sagar@kali)-[~/Desktop/memory_analysis]
$ sudo vol -f c2.img windows.psxview | grep Java
0x84d2c9f8 100.0JavaProgram.ex  2540    True    False   True    True
0x1eb2c9f8      JavaProgram.ex  2540    False   True    False   False
```

## 2.5 Exercise 3

### 2.5.1 Objective

Use the netscan plugin to analyze network activity and extract DLLs and handles.

### 2.5.2 Taks

1. Identify suspicious external IPs and their associated processes.
2. List the DLLs and handles associated with the suspicious process.
3. Write a summary explaining any suspicious activity observed.

### 2.5.3 Results

first we are going to do the identification of suspicious external IPs, it raised suspicion regarding its network activity. Now, the second part is to list the Dlls and handles associated with the suscpicion process.

In this case, the process is **JavaProgram.exe** and these are the dlls:

- *C : \Windows\system32\kerne132.dll*
- *C : \Windows\system32\ADVAP132.dll*
- *C : \Windows\system32\USER32.dll*
- *C : \Windows\system32\msvcrt.dll*
- *C : \Windows\system32\sechost.dll*
- *C : \Windows\system32\RPCRT4.dll*

```

(widza@kali) ~/Desktop/volatility/volatility$
~4 python3 vol.py -f /cz.img dlllist --pid=2548
Volatility 3 Framework 2.11.0
Progress: 100.00

```

PID	Process Base	Size	POB	Name	Path	Finished	LoadTime	File output
2548	JavaProgram.exe	0x400000	0x859c	JavaProgram.exe	C:\Users\IUser\Desktop\JavaProgram.exe	N/A	Disabled	
2548	JavaProgram.exe	0x7110000	0x162800	ntdll.dll	C:\Windows\System32\ntdll.dll	N/A	Disabled	
2548	JavaProgram.exe	0x75510000	0x050800	kernel32.dll	C:\Windows\System32\kernel32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75080000	0x400800	KERNELBASE.dll	C:\Windows\System32\KERNELBASE.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76700000	0x210800	ADVAPI32.dll	C:\Windows\System32\ADVAPI32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75610000	0x4c8000	msvcrt.dll	C:\Windows\System32\msvcrt.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75510000	0x108000	sechost.dll	C:\Windows\System32\sechost.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76010000	0x220800	RPCRT4.dll	C:\Windows\System32\RPCRT4.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76080000	0x4e0800	GDI32.dll	C:\Windows\System32\GDI32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x764c0000	0x090800	USER32.dll	C:\Windows\System32\USER32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x756c0000	0x400800	LPK.dll	C:\Windows\System32\LPK.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76990000	0x900800	USP10.dll	C:\Windows\System32\USP10.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76700000	0x1f0800	IMM32.dll	C:\Windows\System32\IMM32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75990000	0x0c0800	MSCTF.dll	C:\Windows\System32\MSCTF.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75010000	0x4c4000	SHELL32.dll	C:\Windows\System32\SHELL32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76990000	0x570800	SHLAPI.dll	C:\Windows\System32\SHLAPI.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76e80000	0x2ab000	WININET.dll	C:\Windows\System32\WININET.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75240000	0x400800	api-ms-win-downlevel-user32-l1-1-0.dll	C:\Windows\System32\api-ms-win-downlevel-user32-l1-1-0.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75240000	0x400800	api-ms-win-downlevel-shlwapi-l1-1-0.dll	C:\Windows\System32\api-ms-win-downlevel-shlwapi-l1-1-0.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75330000	0x400800	api-ms-win-downlevel-version-l1-1-0.dll	C:\Windows\System32\api-ms-win-downlevel-version-l1-1-0.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x74360000	0x090800	version.dll	C:\Windows\System32\version.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75210000	0x100800	api-ms-win-downlevel-normaliz-l1-1-0.dll	C:\Windows\System32\api-ms-win-downlevel-normaliz-l1-1-0.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75680000	0x300800	normaliz.dll	C:\Windows\System32\normaliz.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x750c0000	0x235800	iertutil.dll	C:\Windows\System32\iertutil.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75300000	0x500800	api-ms-win-downlevel-advapi32-l1-1-0.dll	C:\Windows\System32\api-ms-win-downlevel-advapi32-l1-1-0.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75080000	0x170800	USERENV.dll	C:\Windows\System32\USERENV.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75080000	0x090800	profapi.dll	C:\Windows\System32\profapi.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x76990000	0x150800	WS2_32.dll	C:\Windows\System32\WS2_32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x77280000	0x060800	NSI.dll	C:\Windows\System32\NSI.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x75600000	0x150800	ole32.dll	C:\Windows\System32\ole32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x74ec0000	0x100800	Sspicli.dll	C:\Windows\System32\Sspicli.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x71d00000	0x130800	avicap32.dll	C:\Windows\System32\avicap32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x73300000	0x120800	ATMIME.dll	C:\Windows\System32\ATMIME.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x71d00000	0x210800	MSVFW32.dll	C:\Windows\System32\MSVFW32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x70300000	0x400800	COMCTL32.dll	C:\Windows\System32\comctl32.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x74a00000	0x3c0800	mswsock.dll	C:\Windows\System32\mswsock.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x74400000	0x500800	whrtcpip.dll	C:\Windows\System32\whrtcpip.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x74010000	0x400800	uxtheme.dll	C:\Windows\System32\uxtheme.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x73080000	0x130800	dmapi.dll	C:\Windows\System32\dmapi.dll	2019-04-25 20:40:33.000000 UTC	Disabled	
2548	JavaProgram.exe	0x74700000	0x400800	CRYPTBASE.dll	C:\Windows\System32\CRYPTBASE.dll	2019-04-25 20:40:33.000000 UTC	Disabled	

## dlls

Now, let us explain what was happening, it seems like the combination of the execution of JavaProgram.exe and the loaded Dlls implies a likelihood of malicious program. of course this is supported by the evidence that the program was trying/attempting to establish persistent connections to unidentified external IPs.

## Chapter 3

# Dshell

### 3.1 Introduction

Dshell is an extensible network forensic analysis framework. This tool enables rapid development of plugins to support the dissection of network packet captures.

It is reasonable to mention that devcom list some key features:

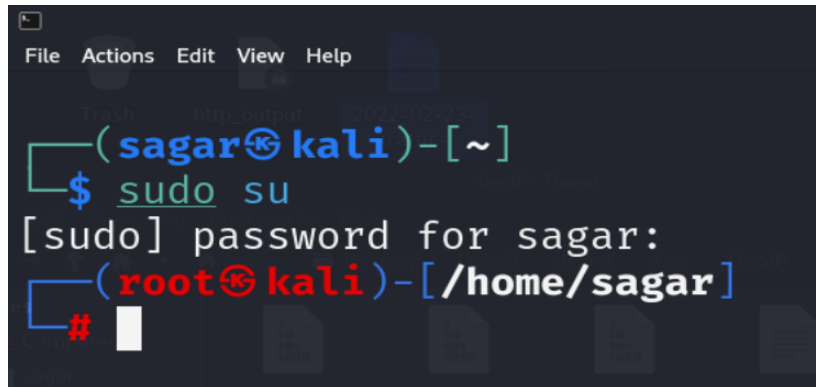
- Deep packet analysis using specialized plugins
- Robust stream reassembly
- IPv4 and IPv6 support
- Multiple user-selectable output formats and the ability to create custom output handlers
- Chainable plugins
- Parallel processing option to divide the handling of data source into separate Python processes
- Enables development of external plugin packs to share and install new externally developed plugins without overlapping the core Dshell plugin directories

### 3.2 Installation

The installation of this tool is quite different from the one on the presentations, first we have to start by cloning the main repository DSHELL REPO. After that we have to go to the terminal and type

```
1 sudo su
```

This is to put the terminal on root mode, of course it is reasonable to mention that we have to input the credentials of the user to get root privileges. the result should look something like this:

A terminal window with a dark background and light-colored text. The window has a menu bar at the top with 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal shows the prompt '(sagar@kali)-[~]' in blue. The user enters '\$ sudo su' in green. The terminal then prompts '[sudo] password for sagar:' in white. The user enters their password (not visible). The terminal then shows the prompt '(root@kali)-[/home/sagar]' in red, indicating root access. A red '#' symbol is visible at the end of the line.

The next step is to install the Dshell with the following command:

```
1 python3 -m pip install Dshell
```

This should be enough for the instalation. In the end, to start the Dshell it is sufficient to mention that we have to type ***dhsell*** on the terminal.

### 3.2.1 Usage

For the usage, the commands has change a little bit, for instance, the PDF from the instructor recommends to use the following command:

```
1 dshell http -r example.pcap --output http output.json
```

For further use of this tool, here is the reference manual that is provided by devcom on how to use Dshell. [MANUAL](#)

As an extra, if desirable, the user can create its own set of features or contribute to the project, here is the Dshell Developer Guide

## 3.3 Disclaimer

For this tool, we reused one of the PCAPs that we had in the beginning of the semester, of course it is reasonable to mention that we already know the output, but it is interesting how it behaves with another tool that also accepts PCAPS

## 3.4 Exercise 1: Analyze HTTP Traffic

### 3.4.1 Objective

The objective of exercise 1 is to use the http decoder in Dshell to identify and extract HTTP communications from a PCAP file.

### 3.4.2 Task

These are the tasks to perform for the first exercise:

1. Use the HTTP decoder to analyze a sample PCAP file.
2. Open the generated Json file and review the HTTP requests and responses.
3. Look for any suspicious URLs, unusual user agents, or large payloads.
4. identify the most frequently requested URL from PCAP.
5. Note the user agents used in the HTTP requests and look for anomalies.
6. Summarize any suspicious findings, such as unexpected data uploads or malicious URLs.

### 3.4.3 Results

Well, First we are going to identify the most frequently requested URL from the PCAP.

```
Could not find GeoIP data files! Country and ASN lookups will not be possible. Check README for instructions on where to find and install necessary data files.
[tls] 2024-07-29 20:38:49 172.16.1.66:49690 -- 20.189.173.16:443 ** mobile.events.data.microsoft.com / **
[tls] 2024-07-29 20:38:56 172.16.1.66:49708 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:57 172.16.1.66:49709 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:57 172.16.1.66:49710 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:57 172.16.1.66:49711 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:58 172.16.1.66:49712 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:58 172.16.1.66:49713 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:58 172.16.1.66:49714 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:59 172.16.1.66:49715 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:59 172.16.1.66:49716 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:38:59 172.16.1.66:49717 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:39:14 172.16.1.66:49732 -- 23.52.9.222:443 ** go.microsoft.com / **
[tls] 2024-07-29 20:39:14 172.16.1.66:49731 -- 13.69.239.79:443 ** v20.events.data.microsoft.com / events.endpoint.security.microsoft.us,events.endpoint.security.microsoft.us,events.data.microsoft.us,vortex-wln.data.microsoft.com,events.data.microsoft.us,vortex.data.microsoft.com,events.data.microsoft.com,vortex-wln.data.microsoft.com **
[tls] 2024-07-29 20:40:11 172.16.1.66:49756 -- 52.191.219.104:443 ** settings.wln.data.microsoft.com **
[tls] 2024-07-29 20:39:15 172.16.1.66:49733 -- 20.96.153.111:443 ** arc.msn.com / dynmsg.mdnpln.com,fd.apl.lrls.microsoft.com,fd.mit.apl.personalization.ideas.microsoft.com,arc-east.msn.com,arc-apac.msn.com,fd-emea.apl.lrls.microsoft.com,arcproxy.microsoft.com,dynmsg.mdnpln.com,fd.apl.orgmsg.microsoft.com,arc-west.msn.com,arc.msn.com,arc-emea.msn.com **
[tls] 2024-07-29 20:40:41 172.16.1.66:49786 -- 20.166.2.191:443 ** msedge.apl.cdp.microsoft.com / apl.cdp.microsoft.com,*.apl.cdp.microsoft.com,*.backend.apl.cdp.microsoft.com **
[tls] 2024-07-29 20:40:42 172.16.1.66:49787 -- 13.107.42.16:443 ** config.edge.skype.com / **
[tls] 2024-07-29 20:39:17 172.16.1.66:49735 -- 52.109.6.53:443 ** officeclient.microsoft.com / *.officebeta.microsoft.com,officeclient.microsoft.com,office1client.microsoft.com,config.office.microsoft.com,*.config-df.officeapps.live.com,*.officeapps.live.com,config-df.officeapps.live.com **
[tls] 2024-07-29 20:39:17 172.16.1.66:49736 -- 204.79.197.203:443 ** www.msn.com / **
[tls] 2024-07-29 20:39:23 172.16.1.66:49738 -- 52.109.0.142:443 ** odc.officeapps.live.com / *.odcsn.officeapps.live.com,*.odc-df.officeapps.live.com,*.odc-eu.officeapps.live.com,*.discover.y.apl.office.net,*.hrd.svc.cloud.microsoft,*.odc.officeapps.live.com,*.odc.officeapps.live.com,*.odc.officeapps.live.com,*.hrd.office.com,*.odc-df.officeapps.live.com,*.hrd.svc.cloud.microsoft,*.hrd.office.com,*.officeapps.live.com **
[tls] 2024-07-29 20:42:36 172.16.1.66:49803 -- 20.189.173.26:443 ** v10.events.data.microsoft.com / events.endpoint.security.microsoft.us,events.endpoint.security.microsoft.us,events.data.microsoft.us,vortex-wln.data.microsoft.com,events.data.microsoft.us,vortex-wln.data.microsoft.com,events.data.microsoft.com,vortex-wln.data.microsoft.com **
[tls] 2024-07-29 20:44:16 172.16.1.66:49810 -- 204.79.197.203:443 ** www.msn.com / **
[tls] 2024-07-29 20:44:17 172.16.1.66:49813 -- 204.79.197.203:443 ** apl.msn.com / **
[tls] 2024-07-29 20:39:13 172.16.1.66:49730 -- 23.198.7.175:443 ** www.bing.com / **
[tls] 2024-07-29 20:39:17 172.16.1.66:49734 -- 23.198.7.175:443 ** www.bing.com / **
[tls] 2024-07-29 20:39:18 172.16.1.66:49737 -- 52.109.0.142:443 ** odc.officeapps.live.com / *.odcsn.officeapps.live.com,*.odc-df.officeapps.live.com,*.odc-eu.officeapps.live.com,*.discover.y.apl.office.net,*.hrd.svc.cloud.microsoft,*.odc.officeapps.live.com,*.odc.officeapps.live.com,*.odc.officeapps.live.com,*.hrd.office.com,*.odc-df.officeapps.live.com,*.hrd.svc.cloud.microsoft,*.hrd.office.com,*.officeapps.live.com **
[tls] 2024-07-29 20:39:23 172.16.1.66:49739 -- 20.241.44.114:443 ** g.live.com / bat.bing.com,g.live.com,analyticpixel.microsoft.com,bat.r.msn.com,g.bing.com,unvalidated.g.msn.com,us-central.g.microsoft.com,us-east.g.microsoft.com,g.bing.com,asia.g.microsoft.com,g.msn.co.jp,g.live.co.jp,azureid.microsoft.com,us-west.g.microsoft.com,g.msn.co.uk,w.g.msn.com,g.msn.com,g.microsoftonline.com,europe.g.microsoft.com,g.microsoft.com,g.live.co.uk **
[tls] 2024-07-29 20:39:23 172.16.1.66:49740 -- 23.52.9.149:443 ** oneclnt.sfx.ms / *.sfx.ms,wildcard.sfx.ms **
[tls] 2024-07-29 20:39:24 172.16.1.66:49741 -- 20.7.2.167:443 ** client.wns.windows.com / *.wns.windows.com **
[tls] 2024-07-29 20:39:24 172.16.1.66:49742 -- 23.48.203.208:443 ** assets.msn.com / **
[tls] 2024-07-29 20:39:24 172.16.1.66:49743 -- 23.48.203.208:443 ** assets.msn.com / **
[tls] 2024-07-29 20:39:27 172.16.1.66:49744 -- 13.107.6.93:443 ** default.asp-tas.com / **
```

Dshell output after running the analysis

As we can see the most request URL in the PCAP was **go.microsoft.com**, from this URL we can assume that the person who was using the network was trying to access their email, after this the user also goes to the settings of it and then logouts this URL is accesses 10 times over the whole PCAP.

Now, we have to note the user agents used in the HTTP requests and look for anomalies. Something interesting that got out attention is that after going through the analysis, we found that a repository from github was cloned, installed and everything that was on that repo was installed. We suspect that of course, something was installed after the user accesses a link that was infected, afterwards a malware was installed to the user's system.

In the end, the most frequently requested URL in the PCAP we had was **go.microsoft.com**, indicating that the user was likely trying to access their email and subsequently checked settings before logging out. In total, the URL was accessed ten times through the capture.

It is worth mentioning, that the installation we had from Dshell was behaving weirdly, for example, we had a problem with the HTTP plugin, this prevented us from collecting that information, and we tried on two more virtual machines and the problem might originate in the installation process or the distribution, because it was tried on Kali linux, ubuntu, and fedora.

## 3.5 Exercise 2: Detect DNS Tunneling

### 3.5.1 Objective

The objective of exercise 2 is to use the dns decoder to identify DNS tunneling or exfiltration attempts.

### 3.5.2 Task

These are the tasks to perform for the second exercise:

1. Use the dns decoder to analyze PCAP file for DNS queries
2. we have to open the csv file and examine the queried domain and reponse sizes.
3. we have to look for patterns suggesting DNS tunneling
4. we have to identify domains with unusually long or complex names.



