Université Sorbonne Paris Nord : IUT de Villetaneuse

Memory Forensics with Volatility

Etudiante : Mame Khady WADE

Professeur : Mr Antonio DA SILVA

RAPPORT MEMORY FORENSICS WITH VOLATILITY

Description du challenge

L'ordinateur de ma sœur est tombé en panne. Nous avons eu beaucoup de chance de récupérer ce fichier mémoire. Votre tâche consiste à récupérer tous les fichiers importants du système. D'après ce dont nous nous avons souvenons, nous avons soudainement vu une fenêtre noire s'afficher avec quelque chose en cours d'exécution. Lorsque le crash s'est produit, elle essayait de dessiner quelque chose. C'est tout ce dont nous nous souvenons au moment du crash.

Pour récupérer les fichiers importants à partir du fichier de vidage mémoire, nous allons utiliser le framework Volatility, qui est conçu pour l'analyse de la mémoire. Pour effectuer une analyse de la mémoire et extraire les fichiers pertinents, la procédure comprend deux parties :

Partie 1 : Installation de volatility

Etape1: Installons les dépendances du système

Pour installer les dépendances du système, nous allons utiliser la commande suivante :

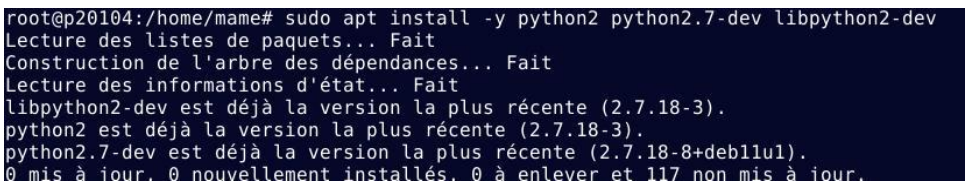
"sudo apt install -y build-essential git libdistorm3-dev yara libraw1394-11 libcapstone-dev" capstone-tool tzdata"

Cette commande sert à préparer un environnement de développement avec des outils et des bibliothèques couramment utilisés pour compiler des logiciels, travailler avec le contrôle de version, effectuer des analyses de désassemblage, et plus encore.

Etape 2: Install pip for Python 2

Pour installer pip pour Python2, nous allons exécuter les commandes suivantes :

- **sudo apt install -y python2 python2.7-dev libpython2-dev**



Cette commande sert à installer Python 2 et les bibliothèques de développement associées sur un système basé sur Debian.

- **Sudo curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py**

```
root@p20104:/home/mame# sudo curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1863k  100 1863k    0     0  9002k      0 --:--:-- --:--:-- --:--:-- 9002k
root@p20104:/home/mame#
```

Cette commande utilise `curl` pour télécharger le script `get-pip.py` depuis l'URL spécifiée et le sauvegarde localement dans le fichier `get-pip.py`.

Après l'exécution de cette commande, le script `get-pip.py` sera présent localement, prêt à être exécuté pour installer `pip` pour Python 2.7.

- **sudo python2 get-pip.py**

```
root@p20104:/home/mame# sudo python2 get-pip.py
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade
ry 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/e
ality.
Collecting pip<21.0
  Using cached pip-20.3.4-py2.py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.3.4
    Uninstalling pip-20.3.4:
      Successfully uninstalled pip-20.3.4
Successfully installed pip-20.3.4
```

Cette commande exécute le script `get-pip.py` à l'aide de Python 2.

- **sudo python2 -m pip install -U setuptools wheel**

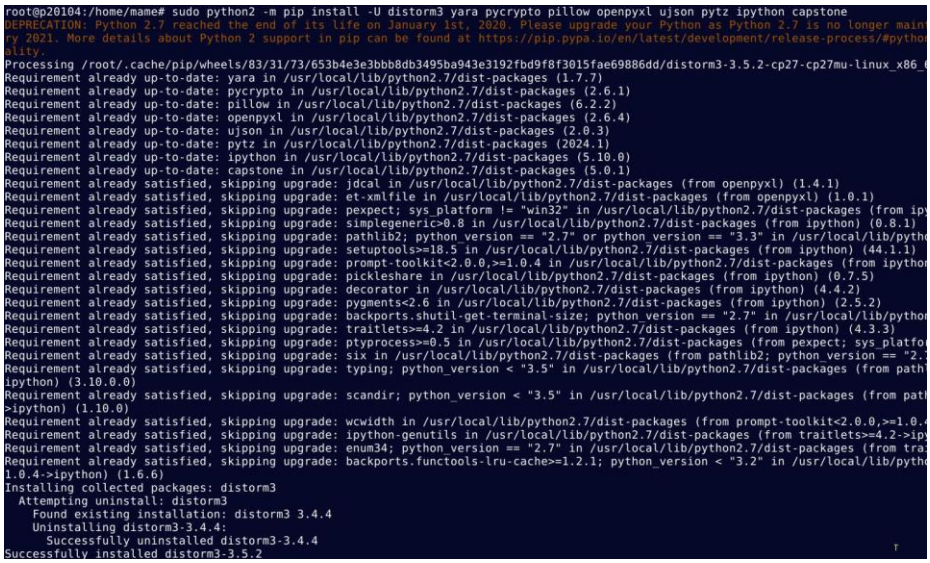
```
root@p20104:/home/mame# sudo python2 -m pip install -U setuptools wheel
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python
ry 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/dev
ality.
Requirement already up-to-date: setuptools in /usr/local/lib/python2.7/dist-packages (44.1.1)
Requirement already up-to-date: wheel in /usr/local/lib/python2.7/dist-packages (0.37.1)
```

Cette commande installe `setuptools` et `wheel` pour Python 2, en utilisant `pip` pour gérer les paquets Python.

Etape3 : Installer volatility 2 et ses dépendances

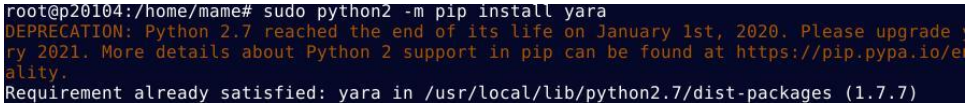
Pour installer à l'échelle du système pour tous les utilisateurs, nous allons utiliser les commandes suivantes :

- **Sudo python2 -m pip install -U distorm3 yara pycrypto pillow openpyxl ujson pytz ipython capstone**



Cette commande est utilisée pour installer ou mettre à jour ces bibliothèques avec la dernière version disponible.

- **sudo python2 -m pip install yara**



Cette commande permet d'installer la bibliothèque YARA, qui est largement utilisée pour la création et l'utilisation de règles de détection de malwares.

- **sudo ln -s /usr/local/lib/python2.7/dist-packages/usr/lib/libyara.so /usr/lib/libyara.so**

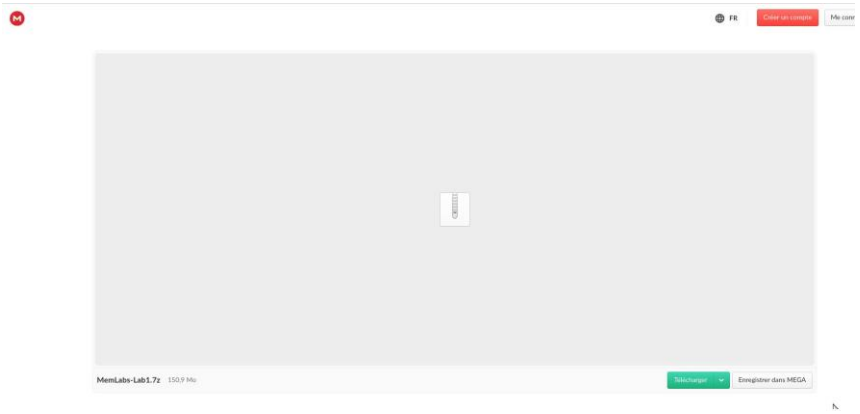
Cette commande installe la version la plus récente de Volatility depuis le dépôt Git officiel de Volatility Foundation.

Partie 2 : Récupérer les fichiers importants à partir du fichier de vidage mémoire.

Pour récupérer les fichiers importants, nous allons suivre les differentes étapes suivantes :

Etape 1 : Télécharger le fichier MemLabs dans le dépôt GitHub :

https://github.com/stuxnet999/MemLabs/tree/master/Lab%201



Etape 2 : Vérifier la version volatility

Nous allons utiliser la dernière version qui est volatility-2.6.1

```
root@p20104:~# vol.py
Volatility Foundation Volatility Framework 2.6.1
```

Etape 3: décompresser les laboratoires mémoire.

Nous allons utiliser la commande suivante :

```
root@p20104:/home/mame# p7zip -d MemLabs-Lab1.7z

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=fr_FR.UTF-8,Utf16=on,HugeFiles=on,64 bits,12 CPUs 12th

Scanning the drive for archives:
1 file, 158197742 bytes (151 MiB)

Extracting archive: MemLabs-Lab1.7z
--
Path = MemLabs-Lab1.7z
Type = 7z
Physical Size = 158197742
Headers Size = 146
Method = LZMA2:24
Solid = -
Blocks = 1


Would you like to replace the existing file:
  Path:     ./MemoryDump_Lab1.raw
  Size:     1073676288 bytes (1024 MiB)
  Modified: 2019-12-11 15:38:17
with the file from archive:
  Path:     MemoryDump_Lab1.raw
  Size:     1073676288 bytes (1024 MiB)
  Modified: 2019-12-11 15:38:17
? (Y)es / (N)o / (A)lways / (S)kip all / A(u)to rename all / (Q)uit? y

Everything is Ok

Size:       1073676288
Compressed: 158197742
```

Etape 4 : Ouvrir volatility

Ici le fichier volatility est nommé vol.py. Nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py --help
Volatility Foundation Volatility Framework 2.6.1
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/mame/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --plugins=PLUGINS     Additional plugin directories to use (colon separate
  --info                Print information about all registered objects
  --cache-directory=/home/mame/.cache/volatility
                        Directory where cache files are stored
  --cache               Use caching
  --tz=TZ               Sets the (Olson) timezone for displaying timestamps
```

La commande fournit des informations sur l'utilisation de base et les options disponibles du framework Volatility.

Etape 5 : liste des profils et des espaces d'adressage disponibles dans volatility.

Nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py --info
Volatility Foundation Volatility Framework 2.6.1


Profiles
--------
VistaSP0x64           - A Profile for Windows Vista SP0 x64
VistaSP0x86           - A Profile for Windows Vista SP0 x86
VistaSP1x64           - A Profile for Windows Vista SP1 x64
VistaSP1x86           - A Profile for Windows Vista SP1 x86
VistaSP2x64           - A Profile for Windows Vista SP2 x64
VistaSP2x86           - A Profile for Windows Vista SP2 x86
Win10x64              - A Profile for Windows 10 x64
Win10x64_10240_17770  - A Profile for Windows 10 x64 (10.0.10240.17770 / 2018-02-10)
Win10x64_10586        - A Profile for Windows 10 x64 (10.0.10586.306 / 2016-04-23)
Win10x64_14393        - A Profile for Windows 10 x64 (10.0.14393.0 / 2016-07-16)
Win10x64_15063        - A Profile for Windows 10 x64 (10.0.15063.0 / 2017-04-04)
Win10x64_16299        - A Profile for Windows 10 x64 (10.0.16299.0 / 2017-09-22)
Win10x64_17134        - A Profile for Windows 10 x64 (10.0.17134.1 / 2018-04-11)
Win10x64_17763        - A Profile for Windows 10 x64 (10.0.17763.0 / 2018-10-12)
Win10x64_18362        - A Profile for Windows 10 x64 (10.0.18362.0 / 2019-04-23)
Win10x64_19041        - A Profile for Windows 10 x64 (10.0.19041.0 / 2020-04-17)
Win10x86              - A Profile for Windows 10 x86
Win10x86_10240_17770  - A Profile for Windows 10 x86 (10.0.10240.17770 / 2018-02-10)
Win10x86_10586        - A Profile for Windows 10 x86 (10.0.10586.420 / 2016-05-28)
Win10x86_14393        - A Profile for Windows 10 x86 (10.0.14393.0 / 2016-07-16)
Win10x86_15063        - A Profile for Windows 10 x86 (10.0.15063.0 / 2017-04-04)
Win10x86_16299        - A Profile for Windows 10 x86 (10.0.16299.15 / 2017-09-29)
Win10x86_17134        - A Profile for Windows 10 x86 (10.0.17134.1 / 2018-04-11)
Win10x86_17763        - A Profile for Windows 10 x86 (10.0.17763.0 / 2018-10-12)
Win10x86_18362        - A Profile for Windows 10 x86 (10.0.18362.0 / 2019-04-23)
Win10x86_19041        - A Profile for Windows 10 x86 (10.0.19041.0 / 2020-04-17)
Win2003SP0x86         - A Profile for Windows 2003 SP0 x86
Win2003SP1x64         - A Profile for Windows 2003 SP1 x64
Win2003SP1x86         - A Profile for Windows 2003 SP1 x86
Win2003SP2x64         - A Profile for Windows 2003 SP2 x64
Win2003SP2x86         - A Profile for Windows 2003 SP2 x86
Win2008R2SP0x64       - A Profile for Windows 2008 R2 SP0 x64
Win2008R2SP1x64       - A Profile for Windows 2008 R2 SP1 x64
Win2008R2SP1x64_23418 - A Profile for Windows 2008 R2 SP1 x64 (6.1.7601.23418 / 2016-04-09)
Win2008R2SP1x64_24000 - A Profile for Windows 2008 R2 SP1 x64 (6.1.7601.24000 / 2016-04-09)
```

Cette commande affiche la liste des profils et des espaces d'adressage disponibles dans l'outil Volatility pour l'analyse des vidages mémoire.

Etape 5 : information sur l'image

Nous pouvons utiliser le plugin d'information sur l'image et cela nous indique la version de Windows.

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win200
8R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7S
P1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/mame/MemoryDump_Lab1.ra
w)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf800028100a0L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff80002811d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2019-12-11 14:38:00 UTC+0000
     Image local date and time : 2019-12-11 20:08:00 +0530
```

Comme nous pouvons le voir, il nous le dit ici, donc les profils suggérés, le premier est à peu près la meilleure option, donc ça ressemble comme ce dump a été exécutant Windows 7 Service Pack 1x64.

Etape 6 : Identifier la structure KDBG (Kernel Debugger Block)

Pour identifier la structure KDBG, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 kdbgscan
Volatility Foundation Volatility Framework 2.6.1
**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800028100a0
Offset (P)                    : 0x28100a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP1x64
Version64                     : 0xf80002810068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.17514.amd64fre.win7sp1_rtm.
PsActiveProcessHead           : 0xfffff80002846b90 (48 processes)
PsLoadedModuleList            : 0xfffff80002864e90 (140 modules)
KernelBase                    : 0xfffff8000261f000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff80002811d00 (CPU 0)
```

Cette commande est utilisée pour identifier la structure KDBG (Kernel Debugger Block) dans le vidage mémoire. La KDBG est une structure de données interne du noyau de Windows, et son analyse peut être utile dans les investigations de la mémoire.

La sortie de la commande indique les détails de deux instanciations potentielles de KDBG dans le vidage mémoire du système, chacune suggérant un profil différent. Dans votre cas, les deux instanciations pointent vers le profil "Win7SP1x64," indiquant que le système exécute Windows 7 Service Pack 1 en version 64 bit.

Etape 7 : Liste des processus en cours de d'exécution dans le vidage mémoire

Pour connaitre les principaux processus présents dans la liste, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ --------
0xfffffa8000ca0040 System                    4      0     80      570 ------      0 2019-12-11 13:41:25 UTC+0000
0xfffffa800148f040 smss.exe                248      4      3       37 ------      0 2019-12-11 13:41:25 UTC+0000
0xfffffa800154f740 csrss.exe               320    312      9      457      0      0 2019-12-11 13:41:32 UTC+0000
0xfffffa8000ca81e0 csrss.exe               368    360      7      199      1      0 2019-12-11 13:41:33 UTC+0000
0xfffffa8001c45060 psxss.exe               376    248     18      786      0      0 2019-12-11 13:41:33 UTC+0000
0xfffffa8001c5f060 winlogon.exe            416    360      4      118      1      0 2019-12-11 13:41:34 UTC+0000
0xfffffa8001c5f630 wininit.exe             424    312      3       75      0      0 2019-12-11 13:41:34 UTC+0000
0xfffffa8001c98530 services.exe            484    424     13      219      0      0 2019-12-11 13:41:35 UTC+0000
0xfffffa8001ca0580 lsass.exe               492    424      9      764      0      0 2019-12-11 13:41:35 UTC+0000
0xfffffa8001ca4b30 lsm.exe                 500    424     11      185      0      0 2019-12-11 13:41:35 UTC+0000
0xfffffa8001cf4b30 svchost.exe             588    484     11      358      0      0 2019-12-11 13:41:39 UTC+0000
0xfffffa8001d327c0 VBoxService.ex          652    484     13      137      0      0 2019-12-11 13:41:40 UTC+0000
```

Cette commande fournit une liste des processus en cours d'exécution dans le vidage mémoire du système.

Ces informations nous ont permis de comprendre quels processus étaient en cours d'exécution au moment du vidage mémoire. Les PID (Process ID) identifient de manière unique chaque processus.

Etape 8 : Afficher une représentation hiérarchique des processus dans le vidage mémoire

Pour afficher une représentation hiérarchique des processus, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xfffffa8000f4c670:explorer.exe                     2504   3000     34    825 2019-12-11 14:37:14 UTC+0000
. 0xfffffa8000f9a4e0:VBoxTray.exe                    2304   2504     14    144 2019-12-11 14:37:14 UTC+0000
. 0xfffffa8001010b30:WinRAR.exe                      1512   2504      6    207 2019-12-11 14:37:23 UTC+0000
 0xfffffa8001c5f630:wininit.exe                       424    312      3     75 2019-12-11 13:41:34 UTC+0000
. 0xfffffa8001c98530:services.exe                     484    424     13    219 2019-12-11 13:41:35 UTC+0000
.. 0xfffffa8002170630:wmpnetwk.exe                   1856    484     16    451 2019-12-11 14:16:08 UTC+0000
.. 0xfffffa8001f91b30:TCPSVCS.EXE                    1416    484      4     97 2019-12-11 13:41:55 UTC+0000
.. 0xfffffa8001da96c0:svchost.exe                     876    484     32    941 2019-12-11 13:41:43 UTC+0000
.. 0xfffffa8001d327c0:VBoxService.ex                  652    484     13    137 2019-12-11 13:41:40 UTC+0000
.. 0xfffffa8000eac770:svchost.exe                    2660    484      6    100 2019-12-11 14:35:14 UTC+0000
.. 0xfffffa80022199e0:svchost.exe                    2368    484      9    365 2019-12-11 14:32:51 UTC+0000
.. 0xfffffa8001e50b30:svchost.exe                    1044    484     14    366 2019-12-11 13:41:48 UTC+0000
.. 0xfffffa8001d8c420:svchost.exe                     816    484     23    569 2019-12-11 13:41:42 UTC+0000
... 0xfffffa80021da060:audiodg.exe                   2064    816      6    131 2019-12-11 14:32:37 UTC+0000
```

Cette commande affiche une représentation hiérarchique des processus dans le vidage mémoire du système.

Cette hiérarchie des processus permet de visualiser les relations entre les différents composants du système au moment du vidage mémoire. Les numéros PID (Process ID) identifient de manière unique chaque processus.

Etape 9 : la ligne de commande associée au processus

Pour afficher la ligne de commande associée au processus, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 cmdline -p 1512
Volatility Foundation Volatility Framework 2.6.1
**************************************************************************
WinRAR.exe pid:   1512
Command line : "C:\Program Files\WinRAR\WinRAR.exe" "C:\Users\Alissa Simpson\Documents\Important.rar"
```

Cette commande affiche la ligne de commande associée au processus WinRAR avec le PID (identifiant de processus) 1512.

Cela montre que le processus WinRAR (PID 1512) a été exécuté avec la ligne de commande spécifiée, qui inclut le chemin vers l'exécutable WinRAR (`C:\Program Files\WinRAR\WinRAR.exe`) ainsi que le chemin du fichier "Important.rar" dans le répertoire "C:\Users\Alissa Simpson\Documents\".

Etape 10 : le Plugin

Le plugin "consoles" est utilisé pour extraire l'historique des commandes en analysant les informations de la console dans la mémoire volatile du système.

```
mame@p20104:~$ vol.py --info | grep consoles
Volatility Foundation Volatility Framework 2.6.1
consoles                - Extract command history by scanning for _CONSOLE_INFORMATION
mame@p20104:~$
```

La ligne que nous avons filtrée avec `grep consoles` dans la sortie de la commande `volatility --info` indique la présence du plugin "consoles" dans la version de Volatility que nous utilisons.

Etape 11 : Extraire les informations sur la ligne de commande associée au processus avec PID 1984.

Pour extraire les informations, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 cmdline -p 1984
Volatility Foundation Volatility Framework 2.6.1
**************************************************
cmd.exe pid:   1984
Command line : "C:\Windows\system32\cmd.exe"
```

Le plugin `cmdline` de Volatility a été utilisé pour extraire les informations sur la ligne de commande associée au processus avec le PID 1984. Dans ce cas, le processus est `cmd.exe`, et la ligne de commande associée est `"C:\Windows\system32\cmd.exe"`.

Etape 12 : Les informations relatives aux consoles en cours d'exécution dans la mémoire.

Pour voir les informations relatives aux consoles en cours d'exécution, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 consoles
Volatility Foundation Volatility Framework 2.6.1
**************************************************
ConsoleProcess: conhost.exe Pid: 2692
Console: 0xff756200 CommandHistorySize: 50
HistoryBufferCount: 1 HistoryBufferMax: 4
OriginalTitle: %SystemRoot%\system32\cmd.exe
Title: C:\Windows\system32\cmd.exe - St4G3$1
AttachedProcess: cmd.exe Pid: 1984 Handle: 0x60
----
CommandHistory: 0x1fe9c0 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x60
Cmd #0 at 0x1de3c0: St4G3$1
----
Screen 0x1e0f70 X:80 Y:300
Dump:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\SmartNet>St4G3$1
ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzMhIX0=
Press any key to continue . . .
**************************************************
ConsoleProcess: conhost.exe Pid: 2260
Console: 0xff756200 CommandHistorySize: 50
HistoryBufferCount: 1 HistoryBufferMax: 4
OriginalTitle: C:\Users\SmartNet\Downloads\DumpIt\DumpIt.exe
Title: C:\Users\SmartNet\Downloads\DumpIt\DumpIt.exe
AttachedProcess: DumpIt.exe Pid: 796 Handle: 0x60
----
CommandHistory: 0x38ea90 Application: DumpIt.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x60
----
Screen 0x371050 X:80 Y:300
Dump:
  DumpIt - v1.3.2.20110401 - One click memory memory dumper
  Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
  Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>


    Address space size:        1073676288 bytes (   1023 Mb)
    Free space size:          24185389056 bytes (  23064 Mb)

    * Destination = \??\C:\Users\SmartNet\Downloads\DumpIt\SMARTNET-PC-20191211-
```

Les résultats fournissent des informations sur les consoles en cours d'exécution, notamment les processus associés, les commandes d'origine, les titres, les processus attachés, l'historique des commandes, et le contenu affiché sur les écrans de ces consoles.

Etape 13 : Décodage

Pour décoder la chaine encodée en base64, nous allons utiliser la commande suivante :

```
mame@p20104:~$ echo "ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzMhIX0=" | base64 -d
mame@p20104:~$ echo "ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzMhIX0=" | base64 -d
flag{th1s_1s_th3_1st_st4g3!!}mame@p20104:~$
```

Etape 14 : Information sur les fichiers présents dans la mémoire du système:

Pour monter des informations sur des fichiers présents dans la mémoire, nous allons exécuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 filescan | grep -i "Important.rar"
Volatility Foundation Volatility Framework 2.6.1
0x000000003fa3ebc0      1      0 R--r-- \Device\HarddiskVolume2\Users\Alissa Simpson\Documents\Important.rar
0x000000003fac3bc0      1      0 R--r-- \Device\HarddiskVolume2\Users\Alissa Simpson\Documents\Important.rar
0x000000003fb48bc0      1      0 R--r-- \Device\HarddiskVolume2\Users\Alissa Simpson\Documents\Important.rar
mame@p20104:~$
```

Les résultats indiquent que le fichier "Important.rar" était présent en mémoire au moment du vidage, et ils fournissent des informations sur son emplacement et ses droits d'accès.

Etape 15 : les Plugins disponibles dans Volatility

Pour afficher les plugins disponibles dans la mémoire, nous allons exécuter la commande :

```
mame@p20104:~$ vol.py --info | grep filescan
Volatility Foundation Volatility Framework 2.6.1
filescan                  - Pool scanner for file objects
mame@p20104:~$
```

La commande `volatility --info` que nous avons exécutée montre les plugins disponibles dans Volatility. La ligne que nous avons filtrée avec `grep filescan` indique la présence du plugin "filescan" dans la version de Volatility que nous utilisons.

Le plugin "filescan" est un scanner de pool pour les objets de fichier.

Etape 16 : Extraire le fichier file.dat

Pour extraire le fichier file.dat à partir de la mémoire dump, nous allons excuter la commande suivante :

```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 dumpfiles -Q 0
x000000003fa3ebc0 -D . file file.dat
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3fa3ebc0   None   \Device\HarddiskVolume2\Users\Alissa Simps
on\Documents\Important.rar
```

Cette commande `vol.py` avec le profil `Win7SP1x64` nous permet d'extraire le fichier "file.dat à partir de la mémoire dump. Il se trouve dans le répertoire courant. Le chemin du fichier dans la mémoire est `\Device\HarddiskVolume2\Users\Alissa Simpson\Documents\Important.rar`.

Etape 19 : Renommer le fichier "file.None.0xfffffa8001034450.dat" à "Important.rar"

Pour renommer le fichier, nous allons exécuter la commande suivante :

```
mame@p20104:~$ mv file.None.0xfffffa8001034450.dat Important.rar
```

Après avoir exécuté cette commande, on remarque que le fichier est renommé de "file.None.0xfffffa8001034450.dat" à "Important.rar".

Etape 20 : Extraire le fichier "file.dat"

Pour extraire le fichier `file.dat`, nous allons utiliser la commande suivante :

```
mame@p20104:~$ tar xzvf Important.tar
tar (child): Important.tar : open impossible: Aucun fichier ou dossier de ce typ
e
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error is not recoverable: exiting now
mame@p20104:~$ ls
 Bureau                                  get-pip.py
'Capture d'écran_2024-02-02_13-48-17.png'   GNS3
'Capture d'écran_2024-02-02_14-04-19.png'   Important.rar
```

Etape 21 : Extrait les hachages de mots de passe

Pour extraire les hachages de mots de passe, nous allons utiliser le hashdump

La commande `hashdump` de Volatility a extrait les hachages de mots de passe (hashes) des comptes utilisateur à partir du dump mémoire. Voici les informations extraites :

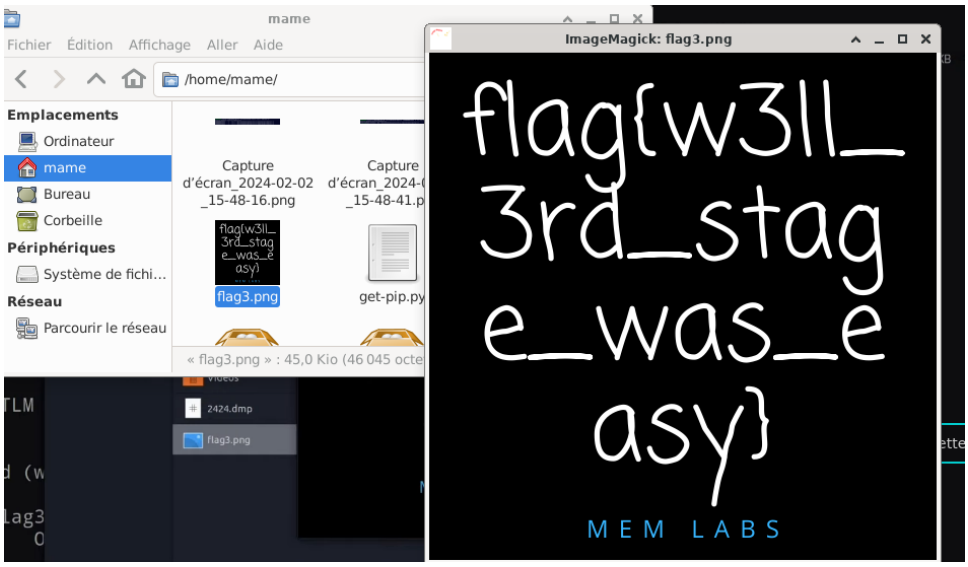
```
mame@p20104:~$ vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 hashdump
Volatility Foundation Volatility Framework 2.6.1
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SmartNet:1001:aad3b435b51404eeaad3b435b51404ee:4943abb39473a6f32c11301f4987e7e0:::
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:f0fc3d257814e08fea06e63c5762ebd5:::
Alissa Simpson:1003:aad3b435b51404eeaad3b435b51404ee:f4ff64c8baac57d22f22edc681055ba6:::
```

Les résultats de cette commande indiquent que le mot de passe requis pour extraire le contenu du fichier Important.rar est le hachage NTLM (en majuscules) du mot de passe du compte Alissa. Nous avons précédemment extrait le hachage NTLM du compte Alissa avec la commande `hashdump`. Utilisons ce hachage NTLM comme mot de passe pour extraire le contenu du fichier Important.rar en exécutant la commande `unrar x Important.rar` et en fournissant le hachage NTLM lorsque vous y êtes invité.

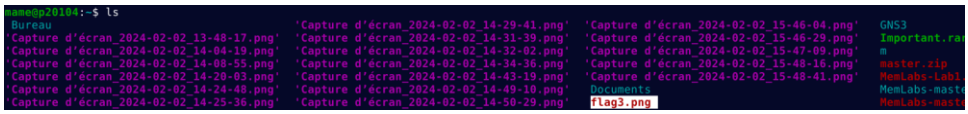
Le mot de passe est: F4FF4C8BAAC57D22F22EDC681055BA6:

```
mame@p20104:~$ unrar x Important.rar

UNRAR 6.00 freeware      Copyright (c) 1993-2020 Alexander Roshal


Extracting from Important.rar

Password is NTLM hash(in uppercase) of Alissa's account passwd.

Enter password (will not be echoed) for flag3.png:

The specified password is incorrect.
Enter password (will not be echoed) for flag3.png:

The specified password is incorrect.
Enter password (will not be echoed) for flag3.png:

The specified password is incorrect.
Enter password (will not be echoed) for flag3.png:

Extracting  flag3.png
All OK
mame@p20104:~$
```

Etape 22 : Flag3.png

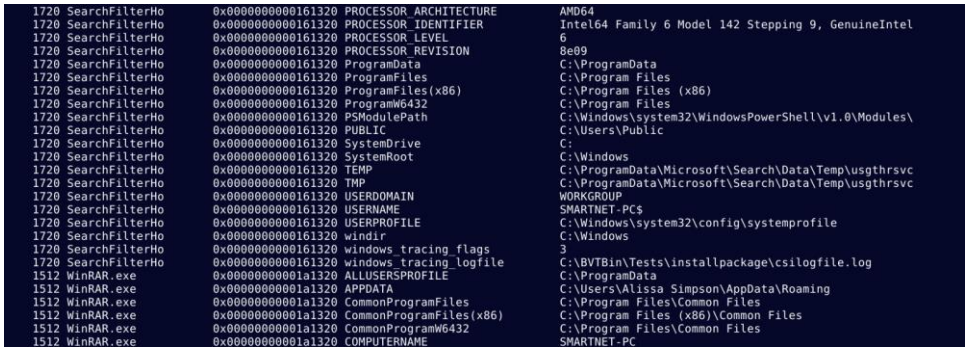


Vérification du fichier flag3.png avec la commande “ls”



Etape 23 : Informations sur les variables d’environnement de deux processus spécifiques.

Pour avoir des informations sur les variables d’environnement de deux processus, nous allons exécuter la commande suivante :

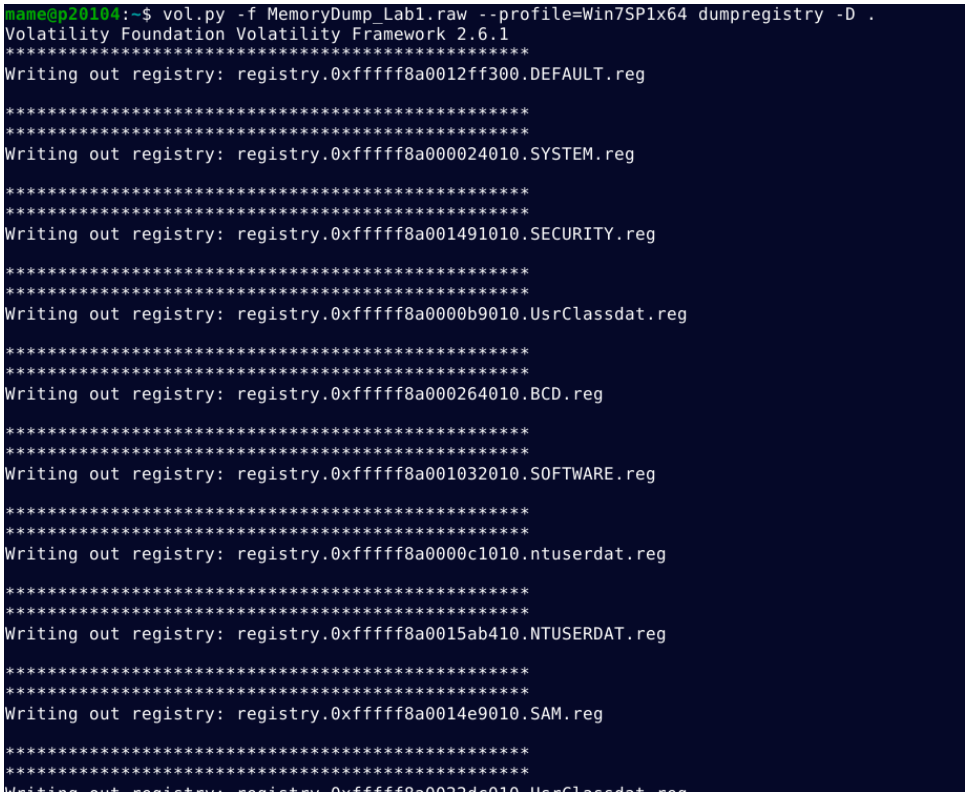


Les résultats extraits à partir de la commande Volatility avec l'option `envars` fournissent des informations sur les variables d'environnement de deux processus spécifiques sur un système Windows 7 SP1 64 bits.

Ces informations nous ont permis d'obtenir un aperçu des paramètres de configuration du système, des spécifications du processeur, des répertoires de programme, des variables d'environnement utilisateur.

Etape 24 : Extraire les hives de la base de registre du système à partir du vidage mémoire.

Pour extraire les hives, nous allons exécuter la commande suivante :



La commande `volatility dumpregistry` nous a permis d'extraire les hives de la base de registre du système à partir du vidage mémoire. Ces hives sont ensuite sauvegardés dans des fichiers séparés.

Ces fichiers contiennent les informations du Registre au moment de la capture mémoire.

Etape 25 : Les fichiers registrent

Pour afficher les fichiers du registre, nous allons utiliser la commande "ls"