



Réponse à incident

Volatility

Wang Peng Chao
IUT de Villetaneuse
Département Réseaux et Télécommunications
2023-2024



SOMMAIRE

Challenge description	2
Partie 1 : Installation de Volatility	2
1.1 Installation des dépendances du système	3
1.2 Installation de pip pour python 2	3
1.3 Installation Volatility 2 et dépendances	4
Partie 2 : Récupération des fichiers importants à partir du fichier de vidage mémoire	5
2.1 Fichier MemLabs du dépôt Github	6
2.2 Décompresser	6
2.3 Vol.py	7
2.4 Décodage	12
2.5 Important.rar	13
2.6 Extraction	13



Challenge description

My sister's computer crashed. We were very fortunate to recover this memory dump. Your job is get all her important files from the system. From what we remember, we suddenly saw a black window pop up with some thing being executed. When the crash happened, she was trying to draw something. Thats all we remember from the time of crash.

Partie 1 : Installation de Volatility

1.1 Installation des dépendances du système

On va maintenant installer les dépendances du système, pour cela il faut utiliser la commande suivante : **"sudo apt install -y build-essential git libdistorm3-dev yara libraw1394-11 libcapstone-dev capstone-tool tzdata"**

```
(root@kali)-[/home/kali]
# sudo apt install -y build-essential git libdistorm3-dev yara libraw1394-11 libcapstone-dev capstone-tool tzdata
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libraw1394-11 is already the newest version (2.1.2-2).
libraw1394-11 set to manually installed.
The following additional packages will be installed:
```

1.2 Installation de pip pour python 2

On va maintenant installer pip pour python 2, pour cela il faut utiliser la commande suivante : **sudo apt install -y python2 python2.7-dev libpython2-dev**

```
(root@kali)-[/home/kali]
# sudo apt install -y python2 python2.7-dev libpython2-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python2 is already the newest version (2.7.18-3).
python2 set to manually installed.
The following additional packages will be installed:
  libpython2.7 libpython2.7-dev
```

Ensuite, il faut télécharger le script "get-pip.py" depuis l'URL et le sauvegarder en local dans le fichier. Pour cela, il faut utiliser la commande suivante :

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



```
(root@kali)-[/home/kali]
# 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  8548k      0 --:--:-- --:--:-- --:--:-- 8587k
```

Nous allons maintenant exécuter le script “**get-pip.py**” pour installer pip pour python 2.
Pour cela, on va utiliser la commande suivante :

sudo python2 get-pip.py

```
(root@kali)-[/home/kali]
# sudo python2 get-pip.py
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python to Python 3+ to avoid this message, which will be removed in a later version of pip.
You can find more information about this warning on https://pip.pypa.io/en/latest/development/release-process/#python-2-deprecation
Collecting pip<21.0
  Downloading pip-20.3.4-py2.py3-none-any.whl (1.5 MB)
    |#####| 1.5 MB 4.0 MB/s
Collecting wheel
  Downloading wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: pip, wheel
Successfully installed pip-20.3.4 wheel-0.37.1
```

Puis, on va installer les outils “setuptools” et “wheel” pour python 2. Pour cela, il faut utiliser la commande suivante :

sudo python2 -m pip install -U setuptools wheel

```
(root@kali)-[/home/kali]
# 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 to Python 3+ to avoid this message, which will be removed in a later version of pip.
You can find more information about this warning on https://pip.pypa.io/en/latest/development/release-process/#python-2-deprecation
Collecting setuptools
  Downloading setuptools-44.1.1-py2.py3-none-any.whl (583 kB)
    |#####| 583 kB 4.8 MB/s
Requirement already up-to-date: wheel in /usr/local/lib/python2.7/dist-packages (0.37.1)
Installing collected packages: setuptools
Successfully installed setuptools-44.1.1
```



1.3 Installation Volatility 2 et dépendances

On va maintenant installer plusieurs paquets à l'échelle du système, ce qui les rend disponibles pour tous les utilisateurs. Pour cela, il faut utiliser la commande suivante :

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

```
(root@kali)-[/home/kali]
# python2 -m pip install -U distorm3 yara pycrypto pillow openpyxl ujson pytz ipython capstone
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as P
in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21
Collecting distorm3
  Downloading distorm3-3.5.2.tar.gz (138 kB)
    | 138 kB 3.8 MB/s
Collecting yara
  Downloading yara-1.7.7.tar.gz (387 kB)
    | 387 kB 42.1 MB/s
Collecting pycrypto
  Downloading pycrypto-2.6.1.tar.gz (446 kB)
```

Ensuite, on va installer la bibliothèque YARA qui permet de faire la création et utilisation des règles de détection de malware. Pour cela, on va utiliser la commande suivante :

sudo python2 -m pip install yara

```
(root@kali)-[/home/kali]
# python2 -m pip install yara
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your
n pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-sup
Requirement already satisfied: yara in /usr/local/lib/python2.7/dist-packages (1.7.7)
```

Puis, on va installer Volatility. Pour cela, il faut utiliser la commande suivante :

python2 -m pip install -U git +<https://github.com/volatilityfoundation/volatility.git>

```
(root@kali)-[/home/kali]
# python2 -m pip install -U git+https://github.com/volatilityfoundation/volatility.git
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop
n pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Collecting git+https://github.com/volatilityfoundation/volatility.git
  Cloning https://github.com/volatilityfoundation/volatility.git to /tmp/pip-req-build-VT3ufz
  Running command git clone -q https://github.com/volatilityfoundation/volatility.git /tmp/pip-req-build-VT3ufz
Building wheels for collected packages: volatility
  Building wheel for volatility (setup.py) ... done
  Created wheel for volatility: filename=volatility-2.6.1-py2-none-any.whl size=6563371 sha256=f5b17af68571a89b5499a69b7156e3e7376259b6bcf5ff45c3596ec8432b4fb9
  Stored in directory: /tmp/pip-ephem-wheel-cache-yF9gW3/wheels/7a/c4/2a/0a32e376b4c5a05335e0659f1633938e1f7ec4b2cd8708b7bc
Successfully built volatility
Installing collected packages: volatility
Successfully installed volatility-2.6.1
```

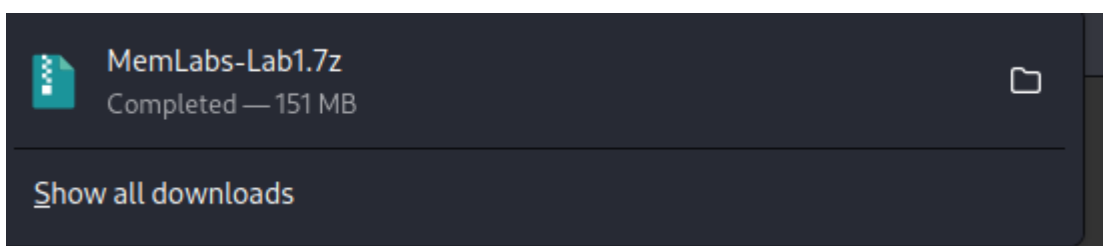
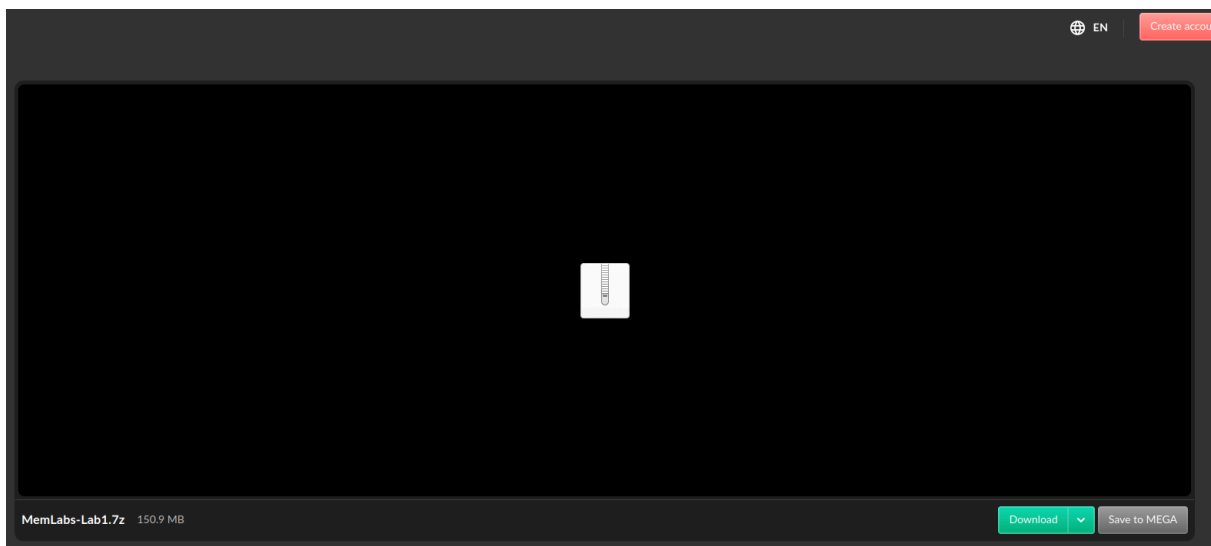


Partie 2 : Récupération des fichiers importants à partir du fichier de vidage mémoire

2.1 Fichier MemLabs du dépôt Github

On va maintenant récupérer les fichiers importants à partir d'un dépôt Github :

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





2.2 Décompresser

On va maintenant décompresser le fichier télécharger. Pour cela, on utilise la commande suivante :

p7zip -d MemLabs-Lab1.7z

```
(root@kali)-[/home/kali]
# 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=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,1 CPU AMD Ryzen 5 5600H with Radeon Graphics (A50F00),ASM,AES-NI)

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

Everything is Ok

Size:      1073676288
Compressed: 158197742
```

2.3 Vol.py

On va maintenant vérifier les options possibles avec le fichier volatility qui est nommé **vol.py**, on peut apercevoir avec l'option **-info** qui est intéressant.

```
# 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=/root/.volatilityrc
                           User based configuration file
  -d, --debug                Debug volatility
  --plugins=PLUGINS          Additional plugin directories to use (colon separated)
  --info                     Print information about all registered objects
  --cache-directory=/root/.cache/volatility
                           Directory where cache files are stored
  --cache                    Use caching
  --tz=TZ                    Sets the (Olson) timezone for displaying timestamps
                           using pytz (if installed) or tzset
  -f FILENAME, --filename=FILENAME
                           Filename to use when opening an image
  --profile=WinXPSP2x86      Name of the profile to load (use --info to see a list
                           of supported profiles)
  -l LOCATION, --location=LOCATION
                           A URN location from which to load an address space
  -w, --write                Enable write support
  --dtb=DTB                  DTB Address
  --output=text              Output in this format (support is module specific, see
                           the Module Output Options below)
  --output-file=OUTPUT_FILE
                           Write output in this file
  -v, --verbose              Verbose information
  --shift=SHIFT              Mac KASLR shift address
  --physical_shift=PHYSICAL_SHIFT
```



En utilisant l'option **-info**, on peut apercevoir la liste des profils et des espaces d'adressage disponible pour permettre d'analyser les vidages mémoires.

```
(root@kali)-[/home/kali]
# vol.py --info
Volatility Foundation Volatility Framework 2.6.1

System
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)
```

Ensuite, on va vérifier les informations de l'image qui permet de savoir la version du Windows. Pour cela, il faut utiliser la commande suivante :

vol.py -f MemoryDump_Lab1.raw image info

```
(root@kali)-[/home/kali]
# 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, Win2008R2SP1x64_24000,
      AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/kali/MemoryDump_Lab1.raw)
      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
```




Puis, on va maintenant identifier le KDBG (Kernel Debugger Block) du win7SP1x64 qui se situe dans le vidage mémoire. En effet, la KDBG est une structure de données interne du noyau de Windows. Son utilité est de pouvoir faire une analyse dans les mémoires. Pour cela, il faut utiliser la commande suivante pour identifier la structure KDBG :

```
(root@kali)-[/home/kali]
# 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        : 0xffffffff80002846b90 (48 processes)
PsLoadedModuleList         : 0xffffffff80002864e90 (140 modules)
KernelBase                 : 0xffffffff8000261f000 (Matches MZ: True)
Major (OptionalHeader)     : 6
Minor (OptionalHeader)     : 1
KPCR                      : 0xffffffff80002811d00 (CPU 0)
```

Offset (V/P) : Les adresses virtuelles et physiques de la structure KDBG.

KDBG owner tag check : Un test pour valider la structure KDBG trouvée.

Profile suggestion (KDBGHeader) : Le profil suggéré basé sur les informations du KDBG.

Version64, Service Pack, Build string : Des détails sur la version du système d'exploitation, y compris le numéro de version, le service pack installé et la chaîne de construction du système.

PsActiveProcessHead, PsLoadedModuleList, KernelBase : Les adresses de certaines structures critiques dans la mémoire du système d'exploitation qui sont nécessaires pour l'analyse forensique.



On peut aussi voir avec l'option **pslist**, une liste des processus en cours d'exécution dans le vidage mémoire du système.

On peut donc apercevoir les processus qui étaient en cours d'exécution lorsque il y a le vidage de mémoire.

```
(root@kali)-[/home/kali]
# 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
-----
0xffffffff8000ca0040 System      4         0        80       570      0      0  2019-12-11 13:41:25 UTC+0000
0xffffffff800148f040 smss.exe    248        4         3        37      0      0  2019-12-11 13:41:25 UTC+0000
0xffffffff800154f740 csrss.exe   320       312         9       457      0      0  2019-12-11 13:41:32 UTC+0000
0xffffffff8000ca81e0 csrss.exe   368       360         7       199      1      0  2019-12-11 13:41:33 UTC+0000
0xffffffff8001c45060 psxs.exe    376       248        18       786      0      0  2019-12-11 13:41:33 UTC+0000
0xffffffff8001c5f060 winlogon.exe 416       360         4       118      1      0  2019-12-11 13:41:34 UTC+0000
0xffffffff8001c5f630 wininit.exe 424       312         3        75      0      0  2019-12-11 13:41:34 UTC+0000
0xffffffff8001c98530 services.exe 484       424        13       219      0      0  2019-12-11 13:41:35 UTC+0000
0xffffffff8001ca0580 lsass.exe   492       424         9       764      0      0  2019-12-11 13:41:35 UTC+0000
0xffffffff8001ca4b30 lsm.exe     500       424        11       185      0      0  2019-12-11 13:41:35 UTC+0000
0xffffffff8001cf4b30 svchost.exe 588       484        11       358      0      0  2019-12-11 13:41:39 UTC+0000
0xffffffff8001d327c0 VBoxService.ex 652       484        13       137      0      0  2019-12-11 13:41:40 UTC+0000
0xffffffff8001d49b30 svchost.exe 720       484         8       279      0      0  2019-12-11 13:41:41 UTC+0000
0xffffffff8001d8c420 svchost.exe 816       484        23       569      0      0  2019-12-11 13:41:42 UTC+0000
0xffffffff8001da5b30 svchost.exe 852       484        28       542      0      0  2019-12-11 13:41:43 UTC+0000
```

Maintenant, on va afficher une représentation hiérarchique des processus dans le vidage mémoire. Pour cela, il faut utiliser la commande suivante :

On peut apercevoir une liste de processus permettant de visualiser les relations entre les différents composants du système au moment du vidage mémoire.

```
(root@kali)-[/home/kali]
# vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 pstree
Volatility Foundation Volatility Framework 2.6.1
Name      Pid      PPid     Thds     Hnds     Time
-----
0xffffffff8000f4c670:explorer.exe      2504    3000        34      825  2019-12-11 14:37:14 UTC+0000
. 0xffffffff8000f9a4e0:VBoxTray.exe    2304    2504        14      144  2019-12-11 14:37:14 UTC+0000
. 0xffffffff8001010b30:WinRAR.exe       1512    2504         6       207  2019-12-11 14:37:23 UTC+0000
0xffffffff8001c5f630:wininit.exe        424     312         3        75  2019-12-11 13:41:34 UTC+0000
. 0xffffffff8001c98530:services.exe      484     424        13       219  2019-12-11 13:41:35 UTC+0000
.. 0xffffffff8002170630:wmpnetwk.exe     1856    484        16       451  2019-12-11 14:16:08 UTC+0000
.. 0xffffffff8001f91b30:TCPSVCS.EXE      1416    484         4        97  2019-12-11 13:41:55 UTC+0000
.. 0xffffffff8001da96c0:svchost.exe       876     484        32       941  2019-12-11 13:41:43 UTC+0000
.. 0xffffffff8001d327c0:VBoxService.ex    652     484        13       137  2019-12-11 13:41:40 UTC+0000
.. 0xffffffff8000eac770:svchost.exe       2660    484         6       100  2019-12-11 14:35:14 UTC+0000
.. 0xffffffff80022199e0:svchost.exe       2368    484         9       365  2019-12-11 14:32:51 UTC+0000
.. 0xffffffff8001e50b30:svchost.exe       1044    484        14       366  2019-12-11 13:41:48 UTC+0000
.. 0xffffffff8001d8c420:svchost.exe        816     484        23       569  2019-12-11 13:41:42 UTC+0000
... 0xffffffff80021da060:audiodg.exe       2064    816         6       131  2019-12-11 14:32:37 UTC+0000
... 0xffffffff8001c38580:svchost.exe        948     484        13       322  2019-12-11 14:16:07 UTC+0000
.. 0xffffffff8001eba230:spoolsv.exe       1208    484        13       282  2019-12-11 13:41:51 UTC+0000
.. 0xffffffff8001d376f0:SearchIndexer.     480     484        14       701  2019-12-11 14:16:09 UTC+0000
... 0xffffffff8000ff630:SearchProtocol    2524    480         7       226  2019-12-11 14:37:21 UTC+0000
```



Ensuite, on va afficher la ligne de commande associée au processus. Pour cela, il faut utiliser la commande suivante :

On peut apercevoir que le programme WinRAR (avec l'ID de processus 1512) a été lancé en utilisant la ligne de commande fournie, qui contient le chemin d'accès à l'application WinRAR (C:\Program Files\WinRAR\WinRAR.exe) et également le chemin vers le fichier nommé "Important.rar" situé dans le dossier "C:\Users\Alissa Simpson\Documents".

```
(root@kali)-[/home/kali]
# 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"
```

Puis, on va utiliser le plugin consoles qui permet d'extraire des commandes en analysant les informations de la console qui se trouve dans la RAM du système. Pour cela, on utilise la commande suivante :

On peut donc apercevoir la présence du plugin consoles dans le Volatility.

```
(root@kali)-[/home/kali]
# vol.py --info | grep consoles
Volatility Foundation Volatility Framework 2.6.1
consoles - Extract command history by scanning for _CONSOLE_INFORMATION
```

Maintenant, on va extraire les informations sur la ligne de commande associée au processus du PID 1984. Pour cela, on utilise la commande suivante :

On peut donc apercevoir que le processus **cmd.exe** associé à la commande **C:\Windows\system32\cmd.exe**

```
(root@kali)-[/home/kali]
# 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"
```



Après on va vérifier les informations relatives aux consoles en cours d'exécution dans la mémoire. Pour cela, on utilise la commande suivante :

On peut donc apercevoir de diverses informations : 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.

```
(root@kali)-[/home/kali]
# 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
ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzZhIX0=
Press any key to continue . . .
*****
```

2.4 Décodage

On va maintenant décoder la chaîne en base64. Pour cela, on utilise la commande suivante:

```
(root@kali)-[/home/kali]
# echo "ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzZhIX0=" | base64 -d
flag{th1s_1s_th3_1st_st4g3 !! }
```



2.5 Important.rar

On va maintenant vérifier si les informations sur les fichiers présents dans la mémoire. Pour cela, il faut utiliser la commande suivante :

On peut apercevoir que le fichier winrar Important.rar était bien présent en mémoire. De plus, on a aussi aperçu son chemin et droit d'accès.

```
(root@kali)-[/home/kali]
# 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
```

2.6 Extraction

On va maintenant extraire le fichier file.dat. Pour cela, on utilise la commande suivante :

On a donc pu extraire le fichier file.dat qui se trouve dans la mémoire dump.

Son chemin est : \Device\HarddiskVolume2\Users\Alissa

Simpson\Documents\Important.rar

```
(root@kali)-[/home/kali]
# vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003fa3ebc0 -D . file file.dat
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3fa3ebc0 None \Device\HarddiskVolume2\Users\Alissa Simpson\Documents\Important.rar
```

Ensuite, on va renommer fichier **file.None.0xfffffa8001034450.dat** en **Important.rar**. Pour cela, on utilise la commande suivante :

```
(root@kali)-[/home/kali]
# mv file.None.0xfffffa8001034450.dat Important.rar
```

Puis, on extrait le fichier file.dat. Pour cela, on utilise la commande suivante :

```
(root@kali)-[/home/kali]
# tar xzvf Important.rar

gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
```

```
(root@kali)-[/home/kali]
# ls
arp-pcap  Documents  get-pip.py  MemoryDump_Lab1.raw  partage  Public  resultat.nmap.gnmap  resultat.nmap.xml  sortie.txt  Videos
Desktop  Downloads  Important.rar  Music  Pictures  report.html  resultat.nmap.nmap  scan.txt  Templates
```



Après, on va extraire les hachages de mots de passe. Pour cela, on utilise la commande suivante :

On peut donc apercevoir qu'il y a différents comptes, parmi ces comptes qu'on retrouve Alissa et son mot de passe.

F4FF4C8BAAC57D22F22EDC681055BA6

```
(root@kali)-[/home/kali]
# 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:::
```

```
(root@kali)-[/home/kali]
# unrar x Important.rar

UNRAR 6.21 freeware      Copyright (c) 1993-2023 Alexander Roshal

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

Extracting from Important.rar

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

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

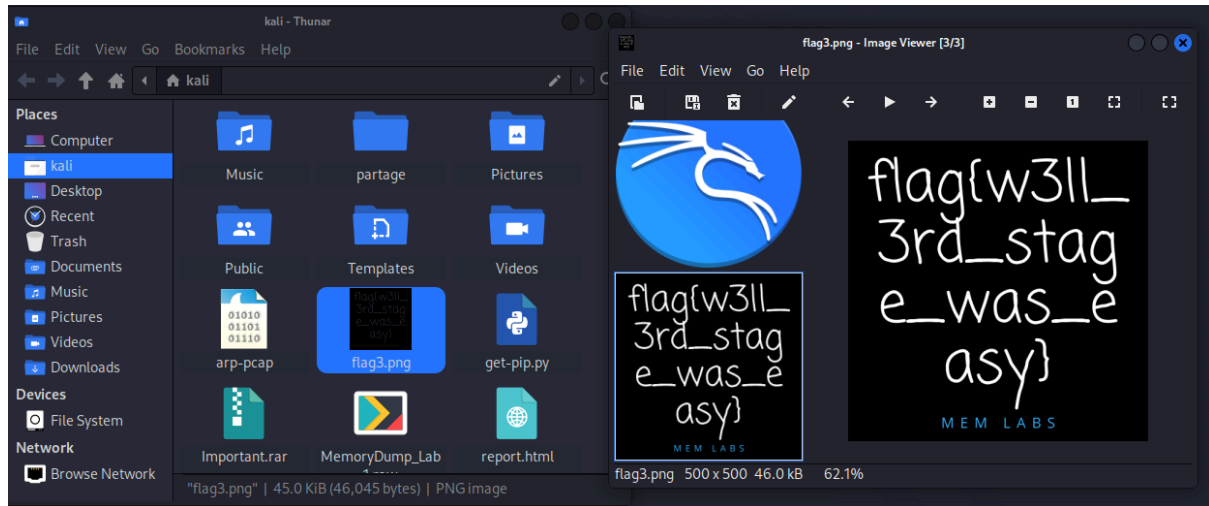
The specified password is incorrect.
Enter password (will not be echoed) for flag3.png:

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

Extracting  flag3.png
All OK
```




On a donc le flag3.png qui se trouve dans /home/kali



```
(root@kali)-[/home/kali]
# ls
arp-pcap  Documents  flag3.png  Important.rar  Music  Pictures  report.html
Desktop  Downloads  get-pip.py  MemoryDump_Lab1.raw  partage  Public  result.txt
```

On va maintenant extraire les hives. Pour cela, on utilise la commande suivante :
Ce sont des informations du registre lors de la capture de la mémoire.

```
(root@kali)-[/home/kali]
# vol.py -f MemoryDump_Lab1.raw --profile=Win7SP1x64 dumphregistry -D.
Volatility Foundation Volatility Framework 2.6.1
*****
Writing out registry: registry.0xfffff8a0012ff300.DEFAULT.reg
*****
*****
Writing out registry: registry.0xfffff8a000024010.SYSTEM.reg
*****
*****
Writing out registry: registry.0xfffff8a001491010.SECURITY.reg
*****
*****
```

Enfin, on va faire un ls pour voir les registres :

```
(root@kali)-[/home/kali]
# ls
arp-pcap  Important.rar  registry.0xfffff8a00000d010.no_name.reg  registry.0xfffff8a001032010.SOFTWARE.reg  registry.0xfffff8a00227a010.ntuserdat.reg  scan.txt
Desktop  MemoryDump_Lab1.raw  registry.0xfffff8a000024010.SYSTEM.reg  registry.0xfffff8a0012ff300.DEFAULT.reg  registry.0xfffff8a0022dc010.UsrClassdat.reg  sortie.txt
Documents  Music  registry.0xfffff8a00004e010.HARDWARE.reg  registry.0xfffff8a001491010.SECURITY.reg  report.html  Templates
Downloads  partage  registry.0xfffff8a0000b9010.UsrClassdat.reg  registry.0xfffff8a0014e9010.SAM.reg  resultat.nmap.gnmap
flag3.png  Pictures  registry.0xfffff8a000c1010.ntuserdat.reg  registry.0xfffff8a0015ab010.NTUSERDAT.reg  resultat.nmap  Videos
get-pip.py  Public  registry.0xfffff8a000264010.BCD.reg  registry.0xfffff8a001626010.NTUSERDAT.reg  resultat.nmap.xml
```

