Réponse à incident

Volatility

Wang Peng Chao

IUT de Villetaneuse
Département Réseaux et Télécommunications
2023-2024

SOMMAIRE

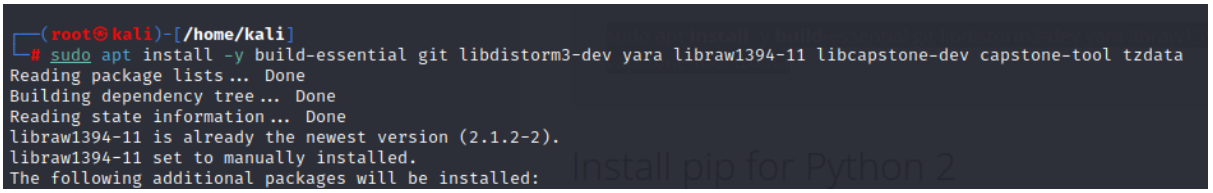
Challenge description

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

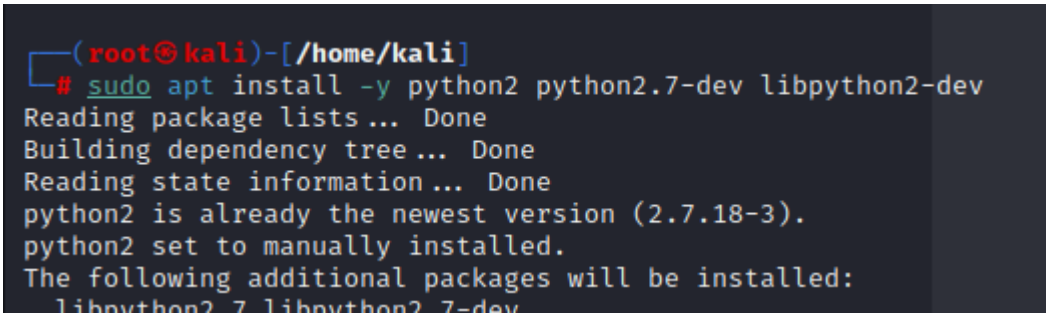
1.1 Installation des dépendances du système

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”**



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**



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

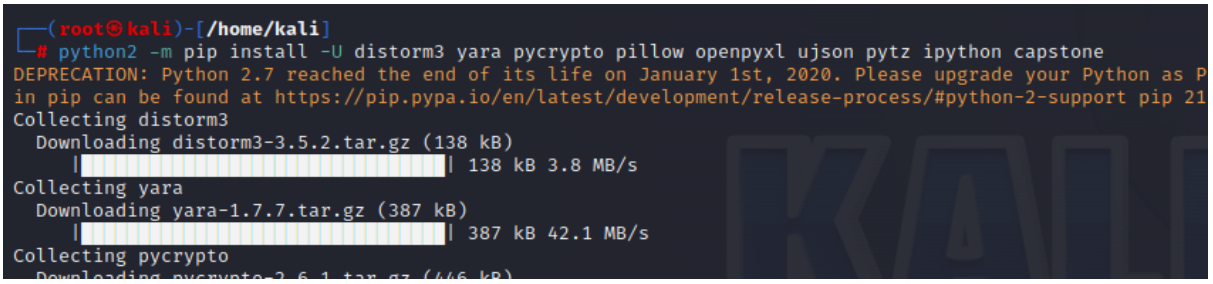
```
┌──(root㉿kali)-[/home/kali]
└─# sudo python2 get-pip.py
DEPRECATION: Python 2.7 reached the end of its life on Janua
in pip can be found at https://pip.pypa.io/en/latest/develop
Collecting pip<21.0
  Downloading pip-20.3.4-py2.py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 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

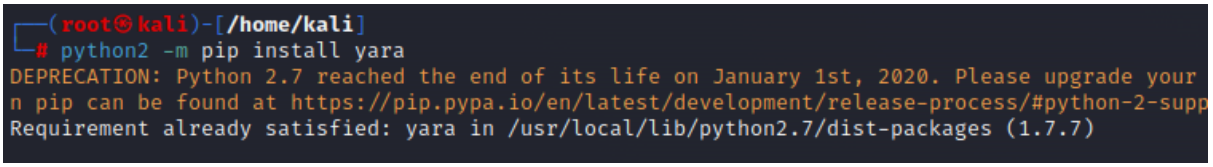
```
┌──(root㉿kali)-[/home/kali]
└─# sudo python2 -m pip install -U setuptools wheel
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade yo
in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-
Collecting setuptools
  Downloading 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

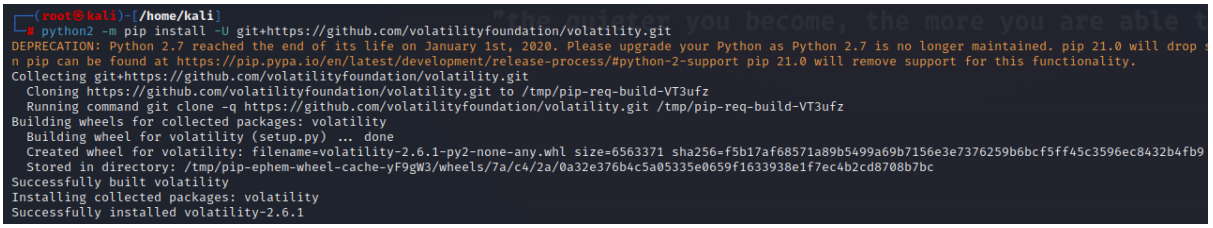
On va maintenant installer plusieurs paquets à l'échelle du système, ce qui les rend disponibles pour tous les utilisateurs. Pour cela, il faut utiliser la commande suivante :
Sudo python2 -m pip install -U distorm3 yara pycrypto pillow openpyxl ujson pytz ipython capstone



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



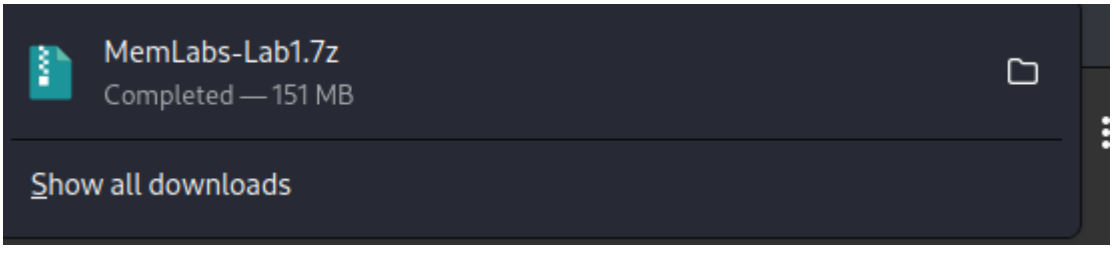
Puis, on va installer Volatility. Pour cela, il faut utiliser la commande suivante :
python2 -m pip install -U git +https://github.com/volatilityfoundation/volatility.git



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

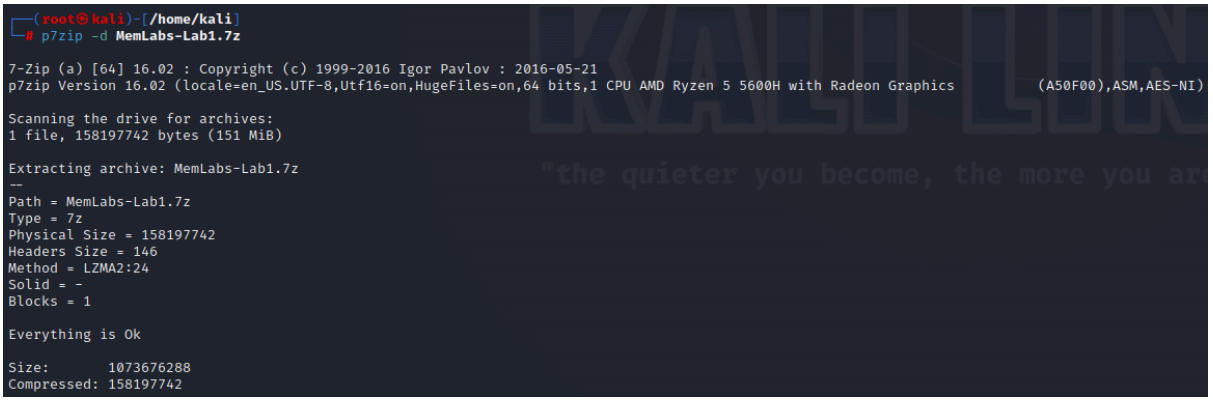
2.1 Fichier MemLabs du dépôt Github

On va maintenant récupérer les fichiers importants à partir d'un dépot Github : https://github.com/stuxnet999/MemLabs/tree/master/Lab%201



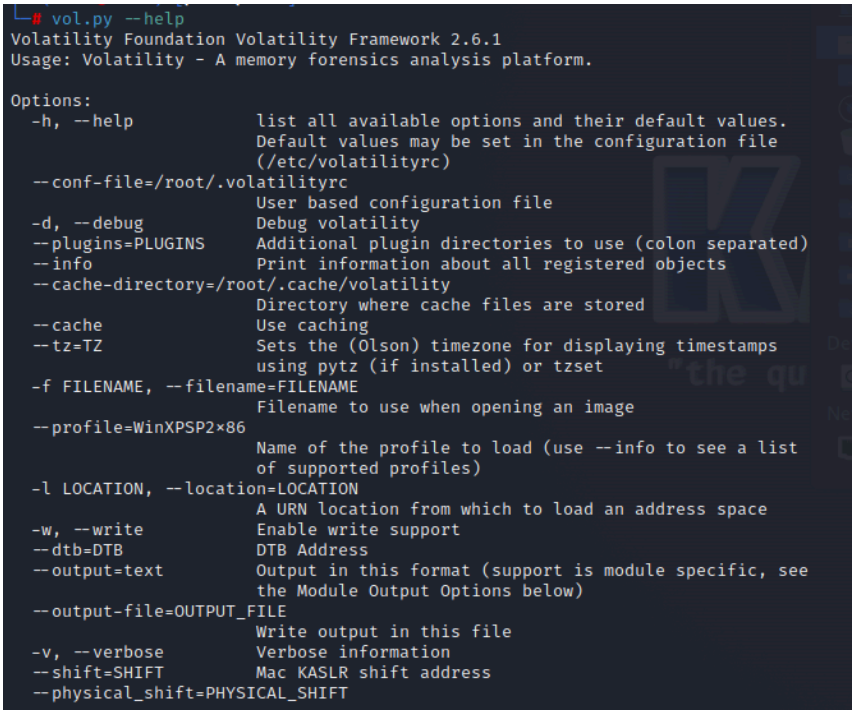
2.2 Décompresser

On va maintenant décompresser le fichier télécharger. Pour cela, on utilise la commande suivante :
p7zip -d MemLabs-Lab1.7z



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.



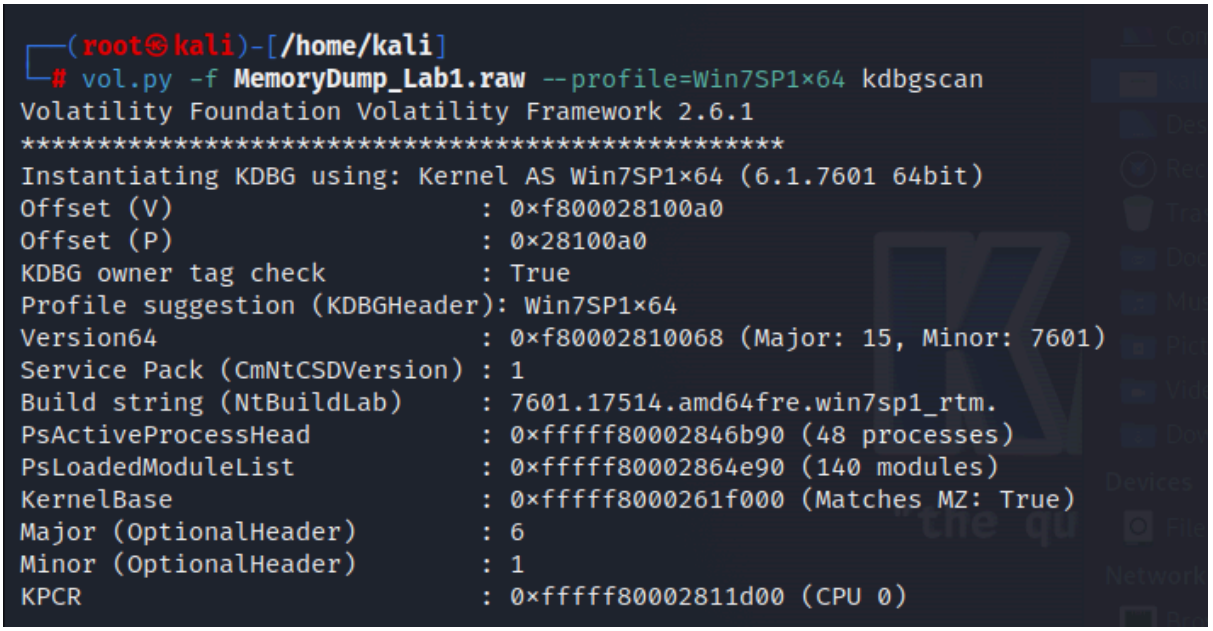
En utilisant l'option **–info**, on peut apercevoir la liste des profils et des espaces d'adressage disponible pour permettre d'analyser les vidages mémoires.

```
┌──(root㉿kali)-[/home/kali]
└─# vol.py --info
Volatility Foundation Volatility Framework 2.6.1

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 :



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
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
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
0xfffffa80154f740 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
0xfffffa8001d49b30 svchost.exe             720    484      8      279      0      0 2019-12-11 13:41:41 UTC+0000
0xfffffa8001d8c420 svchost.exe             816    484     23      569      0      0 2019-12-11 13:41:42 UTC+0000
0xfffffa8001da5b30 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
-------------------------------------------------- ------ ------ ------ ------ ----
 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
.. 0xfffffa8001c38580:svchost.exe                     948    484     13    322 2019-12-11 14:16:07 UTC+0000
.. 0xfffffa8001eba230:spoolsv.exe                    1208    484     13    282 2019-12-11 13:41:51 UTC+0000
.. 0xfffffa8001d376f0:SearchIndexer.                  480    484     14    701 2019-12-11 14:16:09 UTC+0000
... 0xfffffa8000fff630: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 divers 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
ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzMhIX0=
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 "ZmxhZ3t0aDFzXzFzX3RoM18xc3Rfc3Q0ZzMhIX0=" | 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 :

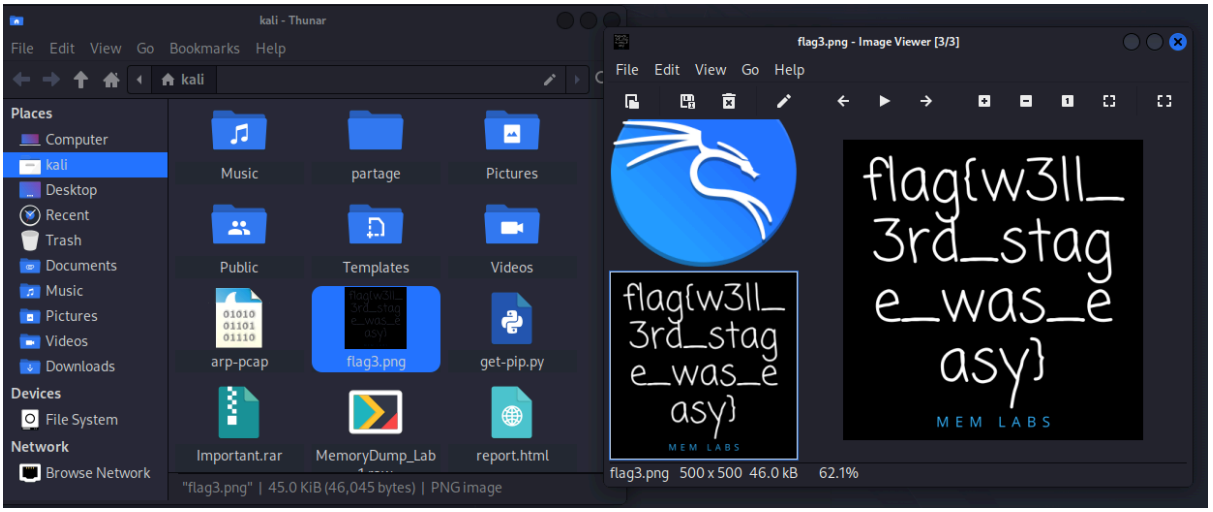
On peut donc apercevoir qu'il y a différents comptes, parmi ces comptes qu'on retrouve Alissa et son mot de passe.

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                                                 OK
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  repor
Desktop   Downloads  get-pip.py MemoryDump_Lab1.raw    partage Public   resul
```

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 dumpregistry -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                         Videos
flag3.png   Pictures               registry.0xfffff8a0000c1010.ntuserdat.reg registry.0xfffff8a0015ab410.NTUSERDAT.reg  resultat.nmap.nmap
get-pip.py  Public                 registry.0xfffff8a000264010.BCD.reg       registry.0xfffff8a001626010.NTUSERDAT.reg  resultat.nmap.xml
```