

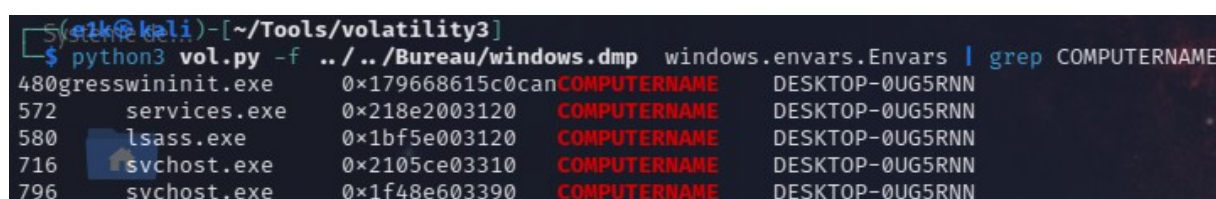
# The crazy admin – Forensic

## Partie 1

Pour cette épreuve, un dump windows, nous est proposé. Nous allons donc devoir utiliser volatility.

Cette première partie consiste à récupérer le nom de l'ordinateur. Pour ce faire, nous devons afficher les variables d'environnement :

```
python3 vol.py -f windows.dmp windows.envvars.Envvars | grep
COMPUTERNAME
```



```
(elk@kali)~[~/Tools/volatility3]
$ python3 vol.py -f ../Bureau/windows.dmp windows.envvars.Envvars | grep COMPUTERNAME
480 gsswininit.exe 0x179668615c0can COMPUTERNAME DESKTOP-0UG5RNN
572 services.exe 0x218e2003120 COMPUTERNAME DESKTOP-0UG5RNN
580 lsass.exe 0x1bf5e003120 COMPUTERNAME DESKTOP-0UG5RNN
716 svchost.exe 0x2105ce03310 COMPUTERNAME DESKTOP-0UG5RNN
796 svchost.exe 0x1f48e603390 COMPUTERNAME DESKTOP-0UG5RNN
```

Figure 1: Variables d'environnement contenant la chaîne COMPUTERNAME

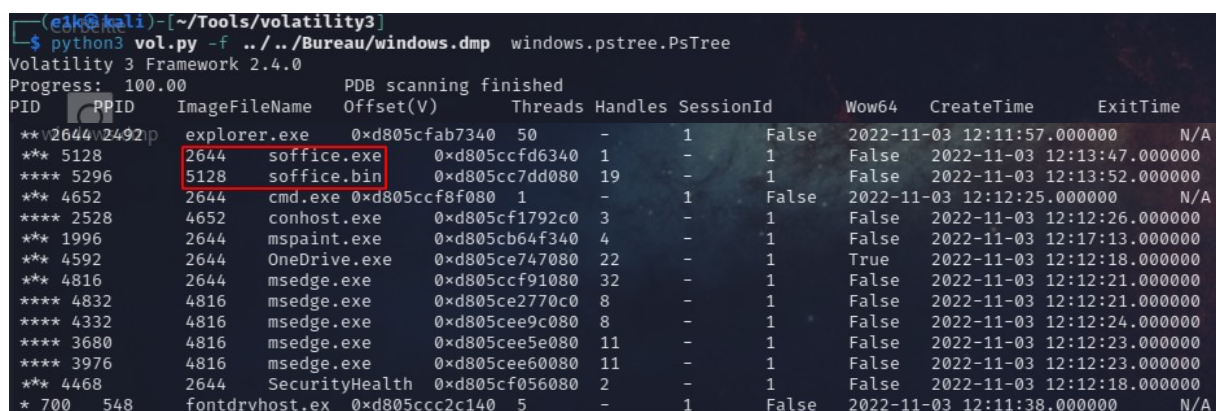
Nous trouvons ainsi le nom de l'ordinateur : DESKTOP-0UG5RNN.

Nous pouvons donc submit le flag ainsi : FMCTF{DESKTOP-0UG5RNN}

## Partie 2

Maintenant, nous devons mettre la main sur le rapport sur lequel l'administrateur travaillait. En affichant l'ensemble des processus, on remarque qu'il travaillait sur office :

```
python3 vol.py -f windows.dmp windows.pstree.PsTree
```



```
(elk@kali)~[~/Tools/volatility3]
$ python3 vol.py -f ../Bureau/windows.dmp windows.pstree.PsTree
Volatility 3 Framework 2.4.0
Progress: 100.00
PDB scanning finished
PID PPID ImageFileName Offset(V) Threads Handles SessionId Wow64 CreateTime ExitTime
** 2644 2492 explorer.exe 0xd805cfab7340 50 - 1 False 2022-11-03 12:11:57.000000 N/A
*** 5128 2644 soffice.exe 0xd805ccfd6340 1 - 1 False 2022-11-03 12:13:47.000000
*** 5296 5128 soffice.bin 0xd805cc7dd080 19 - 1 False 2022-11-03 12:13:52.000000
*** 4652 2644 cmd.exe 0xd805ccf8f080 1 - 1 False 2022-11-03 12:12:25.000000 N/A
**** 2528 4652 conhost.exe 0xd805cf1792c0 3 - 1 False 2022-11-03 12:12:26.000000
*** 1996 2644 mspaint.exe 0xd805cb64f340 4 - 1 False 2022-11-03 12:17:13.000000
*** 4592 2644 OneDrive.exe 0xd805ce747080 22 - 1 True 2022-11-03 12:12:18.000000
*** 4816 2644 msedge.exe 0xd805ccf91080 32 - 1 False 2022-11-03 12:12:21.000000
**** 4832 4816 msedge.exe 0xd805ce2770c0 8 - 1 False 2022-11-03 12:12:21.000000
**** 4332 4816 msedge.exe 0xd805cee9c080 8 - 1 False 2022-11-03 12:12:24.000000
**** 3680 4816 msedge.exe 0xd805cee5e080 11 - 1 False 2022-11-03 12:12:23.000000
**** 3976 4816 msedge.exe 0xd805cee60080 11 - 1 False 2022-11-03 12:12:23.000000
*** 4468 2644 SecurityHealth 0xd805cf056080 2 - 1 False 2022-11-03 12:12:18.000000
* 700 548 fontdrvhost.ex 0xd805ccc2c140 5 - 1 False 2022-11-03 12:11:38.000000 N/A
```

Figure 2: Arbre des processus de l'image windows

elk

On trouve 2 processus correspondant à Office. Essayons donc de dump l'ensemble des fichiers lié à soffice.bin (PID : 5196) :

```
vol3d -f windows.dmp - windows.dumpfiles.DumpFiles --pid 5196
```

```
(elk@kali)~[~/Tools/volatility3]
$ python3 vol.py -f ../Bureau/windows.dmp windows.dumpfiles.DumpFiles --pid 5128
Volatility 3 Framework 2.4.0
Progress: 100.00 PDB scanning finished
Cache fileObject FileName Result
ImageSectionObject 0xd805cb6f3250 KernelBase.dll file.0xd805cb6f3250.0xd805cc829d40.ImageSectionObject.KernelBase.dll.img
ImageSectionObject 0xd805cca9ce70 soffice.exe file.0xd805cca9ce70.0xd805ce56a730.ImageSectionObject soffice.exe.img
ImageSectionObject 0xd805cb2942d0 vcruntime140_1.dll file.0xd805cb2942d0.0xd805ccba5010.ImageSectionObject.vcruntime140_1.dll.img
ImageSectionObject 0xd805cb293970 msvcrt140.dll file.0xd805cb293970.0xd805cca7fcc0.ImageSectionObject.msvcrt140.dll.img
ImageSectionObject 0xd805cb2945f0 vcruntime140.dll file.0xd805cb2945f0.0xd805ccf1dc80.ImageSectionObject.vcruntime140.dll.img
ImageSectionObject 0xd805cb6f30c0 msvcrt140.dll file.0xd805cb6f30c0.0xd805cc829aa0.ImageSectionObject.msvcrt140.dll.img
ImageSectionObject 0xd805ce456de0 apphelp.dll file.0xd805ce456de0.0xd805ce1c7730.ImageSectionObject.apphelp.dll.img
ImageSectionObject 0xd805ccebcb510 user32.dll file.0xd805ccebcb510.0xd805cebedcc0.ImageSectionObject.user32.dll.img
ImageSectionObject 0xd805ccebcb9c0 kernel32.dll file.0xd805ccebcb9c0.0xd805cc55fc60.ImageSectionObject.kernel32.dll.img
ImageSectionObject 0xd805ccebca3e0 ucrtbase.dll file.0xd805ccebca3e0.0xd805cc39ed90.ImageSectionObject.ucrtbase.dll.img
ImageSectionObject 0xd805cb6f4510 gdi32full.dll file.0xd805cb6f4510.0xd805cebe59e0.ImageSectionObject.gdi32full.dll.img
ImageSectionObject 0xd805ccebca250 win32u.dll file.0xd805ccebca250.0xd805cebe6d90.ImageSectionObject.win32u.dll.img
ImageSectionObject 0xd805ccebcbce0 rpcrt4.dll file.0xd805ccebcbce0.0xd805ce85ab90.ImageSectionObject.rpcrt4.dll.img
ImageSectionObject 0xd805ccebca890 SHCore.dll file.0xd805ccebca890.0xd805cc39fc50.ImageSectionObject.SHCore.dll.img
ImageSectionObject 0xd805ccebcb6a0 shell32.dll file.0xd805ccebcb6a0.0xd805ce56ead0.ImageSectionObject.shell32.dll.img
ImageSectionObject 0xd805ccc9b700 combase.dll file.0xd805ccc9b700.0xd805cebf0010.ImageSectionObject.combase.dll.img
ImageSectionObject 0xd805ccc9b3e0 imm32.dll file.0xd805ccc9b3e0.0xd805cebee550.ImageSectionObject.imm32.dll.img
ImageSectionObject 0xd805ccc9ca570 gdi32.dll file.0xd805ccc9ca570.0xd805cebee010.ImageSectionObject.gdi32.dll.img
ImageSectionObject 0xd805ccc9c1f0 msvcrt.dll file.0xd805ccc9c1f0.0xd805cb12d60.ImageSectionObject.msvcrt.dll.img
ImageSectionObject 0xd805cb36c940 ntdll.dll file.0xd805cb36c940.0xd805cb76dca0.ImageSectionObject.ntdll.dll.img
ImageSectionObject 0xd805ccc9b250 sechost.dll file.0xd805ccc9b250.0xd805cebef7f0.ImageSectionObject.sechost.dll.img
```

Figure 3: Dump des fichiers du processus 5128

On ne voit rien. Essayons donc de dump les fichiers liés au deuxième :

```
vol3d -f windows.dmp - windows.dumpfiles.DumpFiles --pid 5296
```

```
(elk@kali)~[~/Tools/volatility3]
$ cat test.txt
Volatility 3 Framework 2.4.0
Cache FileObject FileName Result
DataSectionObject 0xd805cb2c9340 extensions.pmap file.0xd805cb2c9340.0xd805ce15c8b0.DataSectionObject.extensions.pmap.dat
DataSectionObject 0xd805ccce5430 images_colibre.zip Error dumping file
SharedCacheMap 0xd805ccce5430 images_colibre.zip file.0xd805ccce5430.0xd805cf56f260.SharedCacheMap.images_colibre.zip.vacb
DataSectionObject 0xd805ce6492e0 Confidential.odt file.0xd805ce6492e0.0xd805ce15a6f0.DataSectionObject.Confidential.odt.dat
```

Figure 4: Dump des fichiers du processus 5296

Et la bingo ! Confidential.odt.dat

Il nous reste plus qu'à récupérer le fichier, le renommer en Confidential.odt et l'ouvrir. Malheureusement il semble que le document soit corrompu. Toutefois nous pouvons essayer de le réparer avec Libre Office et Bingo ! Voici le flag !

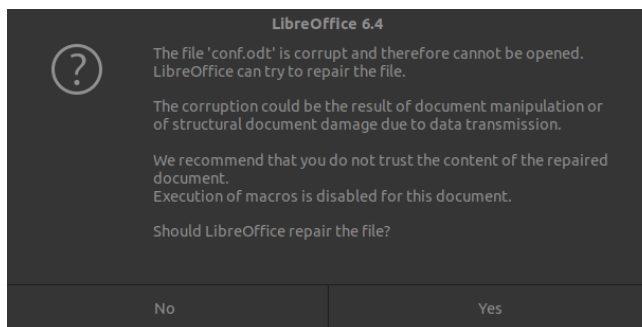


Figure 5: Tentative de récupération



Figure 6: Rapport récupéré sur le dump

Il n'y a plus qu'à remettre tout ça en forme. On peut faire ça en bash : on fait un copier coller du document que l'on affiche puis l'on pipe un `tr` avec l'option `-d` (delete) puis la chaîne à enlever : `\n`.

```
e1k@e1k-EliteBook:~/Downloads$ echo "F
> M
> C
> T
> F
> {
> 7
> h
> 1
> $
> -
> C
> O
> M
> P
> U
> t
> e
> R
> -
> h
> A
> s
> N
> o
> -
> s
> 3
> e
> c
> r
> e
> T
> -
> f
> o
> r
> -
> y
> o
> u
> }" | tr -d "\n"
```

Nous obtenons ainsi le flag : `FMCTF{7h1$_COMPUteR_hAs_No_s3creT_f0r_you}`

### Partie 3

Pour cette troisième partie, on nous indique que l'administrateur est devenu fou et qu'il s'est mis à dessiner... Et il se trouve que nous avons un processus `mspaint`. Nous allons donc devoir dump ce processus et faire du « carving », c'est à dire se ballader dans la mémoire et espérer tomber sur la partie graphique enregistrée dedans. Pour ce faire, nous dumpons la mémoire associée au processus `mspaint.exe` :

```
python3 vol.py -f windows.dmp windows.memmap --pid 6484 --dump
```

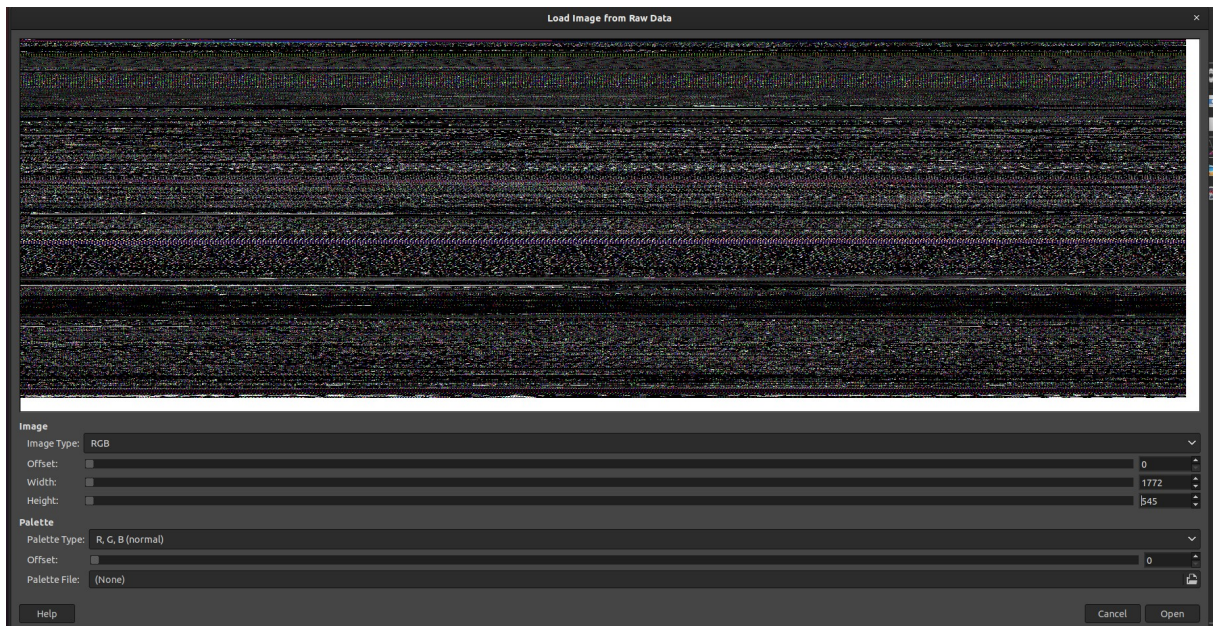


e1k

Nous obtenons ainsi un fichier de 400Mb.

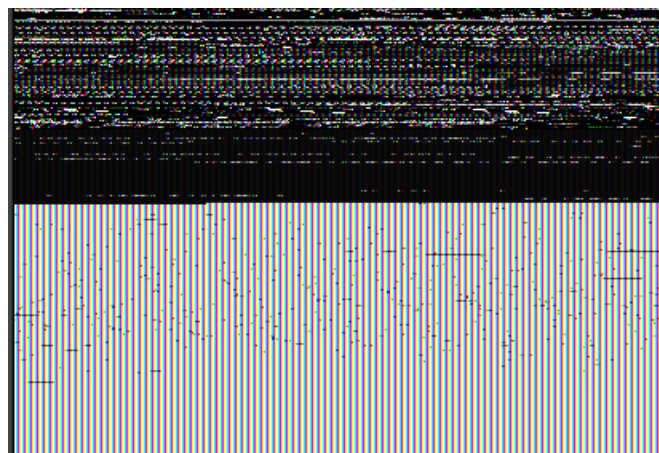
Pour pouvoir « lire » le dump, nous l'ouvrons avec un gimp en changeant le nom de l'extension pour .data.

Ensuite, nous pouvons ajuster l'importer dans gimp et ajuster la taille pour qu'elle occupe la taille de notre écran.



*Figure 7: Visualisation du dump du processus paint*

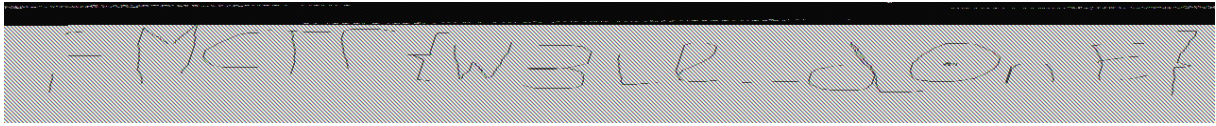
Nous pouvons ensuite nous balader dans la mémoire et on finit par tomber sur une partie interessante : (Offset : 338972156)



*Figure 8: Visualisation de l'état du moteur graphique*

elk

Il ne nous reste plus qu'à ajuster la largeur de l'image (Width : 2683, Height : 587) pour trouver le flag :



*Figure 9: Visualisation du moteur graphique adaptée*

On peut valider le challenge avec : Flag: FMCTF{W3LI\_d0nE}