VINCENT LE TOUX

Les dumps mémoires

Degré de difficulté

Il existe de nombreuses attaques physiques : démontage du disque dur, boot sur un live cd, ... Mais si le disque dur est crypté, toutes ces techniques échouent. Existe-t-il d'autres techniques permettant de contourner les sécurités mises en place ?

La solidité d'une chaîne est conditionnée par celle de son maillon le plus faible. C'est de cette manière que le vol d'un seul ordinateur portable peut entraîner des pertes colossales : des données sensibles sont stockées sur ces PC. Pour renforcer la sécurité, les experts en sécurité recommandent de crypter les disques durs. En effet, les données non cryptées sont facilement accessibles en ayant un accès physique à la machine via diverses techniques telles que le Livecd ou le montage dans un boîtier usb. Cet article va montrer comment ces PC sont protégés et comment il est possible de contourner ces protections en s'attaquant au deuxième élément le plus faible de la chaîne : la mémoire vive.

Protection des PC portables

Si les experts recommandent de protéger les données sensibles, il existe plusieurs façons de le mettre en pratique. Il y a tout d'abord les mots de passe. Plusieurs solutions existent pour les mettre en place : au niveau du BIOS ou encore au niveau du système d'exploitation. Il y a ensuite le cryptage de fichiers individuels. Un programme est installé, comme PGP ou TrueCrypt, et transforme ces fichiers en une suite binaire incompréhensible grâce à un mot de passe ou un dispositif de sécurité. C'est la solution la plus pratique et la plus facile à mettre en place. Néanmoins, elle souffre d'un défaut majeur : elle s'appuie sur le système existant. Rien n'empêche un pirate d'installer un programme surveillant l'appui de touches sur le clavier ; le système d'exploitation doit donc être protégé. La troisième solution consiste à crypter tout le disque dur afin de garantir que le système ne puisse pas être corrompu. Une contrainte : l'authentification doit être réalisée avant le démarrage du système d'exploitation sinon celui-ci ne pourra pas lire les données nécessaires à son lancement. Les utilitaires permettant de crypter un disque dur tels que BitLocker, TrueCrypt, dm-crypt, BitVault, Safeguard, etc mettent en place un environnement dit de Pre Boot Authentication permettant l'authentification avant le démarrage du système d'exploitation. Il s'agit la plupart du temps d'une interface texte type DOS. Une fois le mot de passe saisi, l'outil cherche sur le disque la clé de cryptage elle-même protégée par le mot de passe utilisateur, la décrypte en mémoire et dès lors, lorsqu'un accès au disque est réalisé, décrypte ou encrypte les données à la volée. Ces opérations cryptographiques ne sont pas réalisées par un dispositif spécialisé mais par le processeur et la clé est stockée en mémoire. La Figure 1 récapitule ce processus.

Des données mémorisées

La sécurité des données tel qu'elle est implémentée, c'est-à-dire sans dispositif physique, repose sur le fait que la mémoire vive est effacée immédiatement lorsqu'elle n'est plus sous tension et que la seule façon d'obtenir cette clé est d'avoir

CET ARTICLE EXPLIQUE...

Que de nombreuses données sensibles sont stockées en mémoire.

Comment récupérer ces données.

Comment les analyser.

CE QU'IL FAUT SAVOIR...

Des notions sur l'architecture des ordinateurs.

déjà le contrôle de la machine. Il existe en effet plusieurs façons de réaliser des dumps mémoires et toutes requièrent les droits d'administrateur. La première façon pour déclencher un dump (pour faire du debugging système par exemple) est de provoquer un crash général. En effet le système peut enregistrer toute la mémoire dans le fichier *c:\windows\memory.dmp* avant de s'arrêter. L'option est paramétrable via le *Panneau de configuration*, *Système*, *Avancé*, *Démarrage et récupération* puis *Écrire des informations de débogage*. Il faut alors choisir Image mémoire complète. Un crash peut être forcé avec une combinaison de touches : si la clé *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\i8042prt\Parameters\CrashOnCtrlScroll* vaut 1, l'appui simultané de la touche Control droit et de deux fois la touche Arrêt défil provoque l'arrêt du système (BSOD). La deuxième est bien entendu la mise en veille prolongée. Mais dans le cadre de disques cryptés, ces deux méthodes produisent un dump lui-même crypté. Enfin la dernière méthode utilise le périphérique `\\.\PysicalMemory` permettant de lire directement la mémoire et nécessite pour fonctionner, les droits d'administrateur. Il suffit de télécharger l'outil dd et d'exécuter la commande `dd.exe if=\\.\PhysicalMemory of=mydump.dmp`. Nous venons de le voir, si le disque est crypté ou si une session administrateur n'est pas ouverte, il est impossible de s'attaquer à la mémoire. En pratique donc, la sécurité est suffisante. Les utilisateurs, appartenant en général à une entreprise ne sont jamais connectés comme administrateur, et une fois l'ordinateur éteint, la clé de cryptage ou les mots de passe utilisés sont effacés.

Mémoire à long terme

Pour augmenter l'autonomie, de nombreux dispositifs d'économie d'énergie ont été mis en place. Mise en veille simple ou prolongée, toutes ces astuces permettent de stopper la machine sans devoir la redémarrer. Il suffit de fermer l'écran pour que la machine entre en veille. À l'ouverture de l'écran, le mot de passe Windows est demandé. Un ordinateur dérobé a alors de fortes chances d'être encore allumé mais verrouillé. Est-il impossible alors pour un pirate de forcer le système si le système est verrouillé mais encore allumé sans le redémarrer ? En réalité, la mémoire vive n'est effacée qu'après une ou deux secondes : c'est le phénomène de rémanence. Les chercheurs de l'université de Princeton ont étudié ce dernier et ont publié un article en février 2008 : si la rétention des informations à température ambiante est relativement courte, il a été mis en évidence que celle-ci augmente lorsque la température des modules mémoires diminue. A -50 degrés centigrades, elle peut même atteindre 5 minutes. Attention, car la mémoire n'a pas toute son intégrité et des erreurs apparaissent : approximativement 99% des

Listing 1. *Installation de msramdmp*

```
# effacement de la clé usb
dd if=/dev/zero of=/dev/sda
# création d'une partition FAT16 de 1Mo
# création d'une partition Venix 80286 (type 40)
cfdisk
# formatage (attention au automount)
mkfs.msdos /dev/sda1
# download
wget http://mcgrewsecurity.com/projects/msramdmp/msramdmp.tar.gz
tar xvf msramdmp.tar.gz
wget http://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-3.61.tar.gz
tar xvf syslinux-3.61.tar.gz
# installation
dd if=syslinux-3.61/mbr/mbr.bin of=/dev/sda
syslinux-3.61/unix/syslinux /dev/sda1
mount /dev/sda1 /mnt/sda1
cp msramdmp.c32 syslinux.cfg /mnt/sda1
```

Listing 2. *Écriture de données en mémoire*

```
#!/usr/local/bin/python
a = ""
while 1: a += "HAKIN9"
```

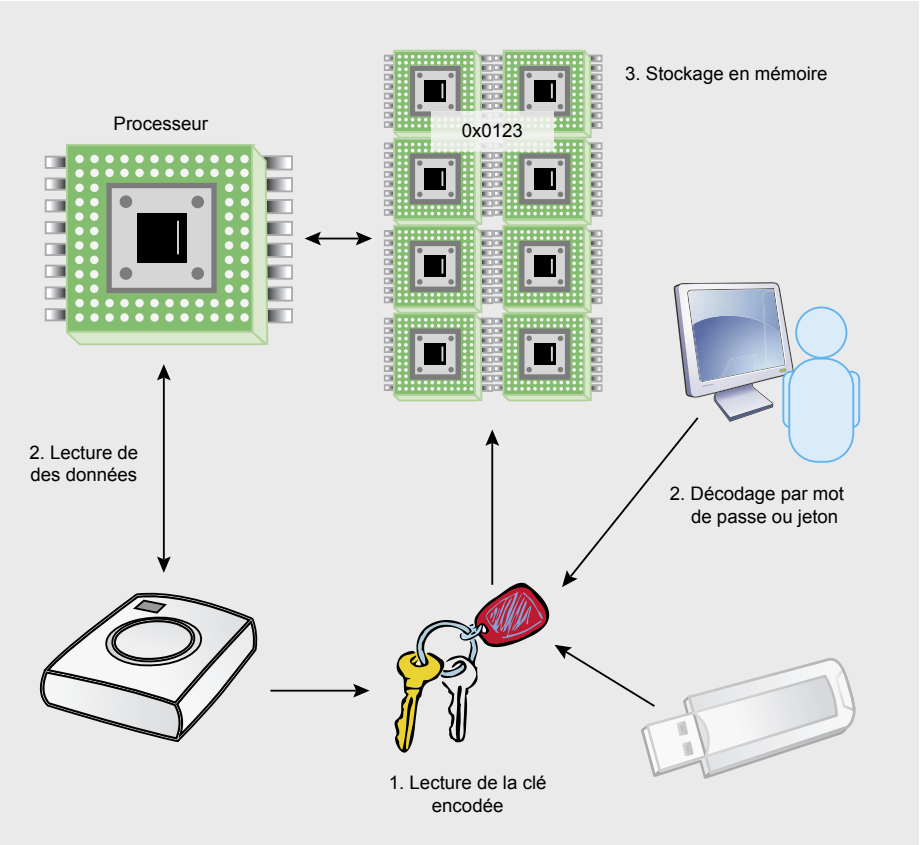

Figure 1. *Utilisation d'un disque crypté*

```
SYSLINUX 3.61 2008-02-03 EBIOS Copyright (C) 1994-2008 H. Peter Anvin

-----------------------------------------------
msramdmp - McGrew Security Ram Dumper - v 0.5
http://mcgrewsecurity.com/projects/msramdmp/
Robert Wesley McGrew: wesley@mcgrewsecurity.com
-----------------------------------------------

Found msramdmp partition at disk 0x80 : partition 2
Partition isn't marked as used.  Using it.
Marked partition as used.
Writing section from 0x00000000 to 0x0009FFFF
Writing section from 0x00100000 to 0x4000000
Done! You can turn off the machine and remove your drive.
boot: _
```

Figure 2. *Msramdmp*

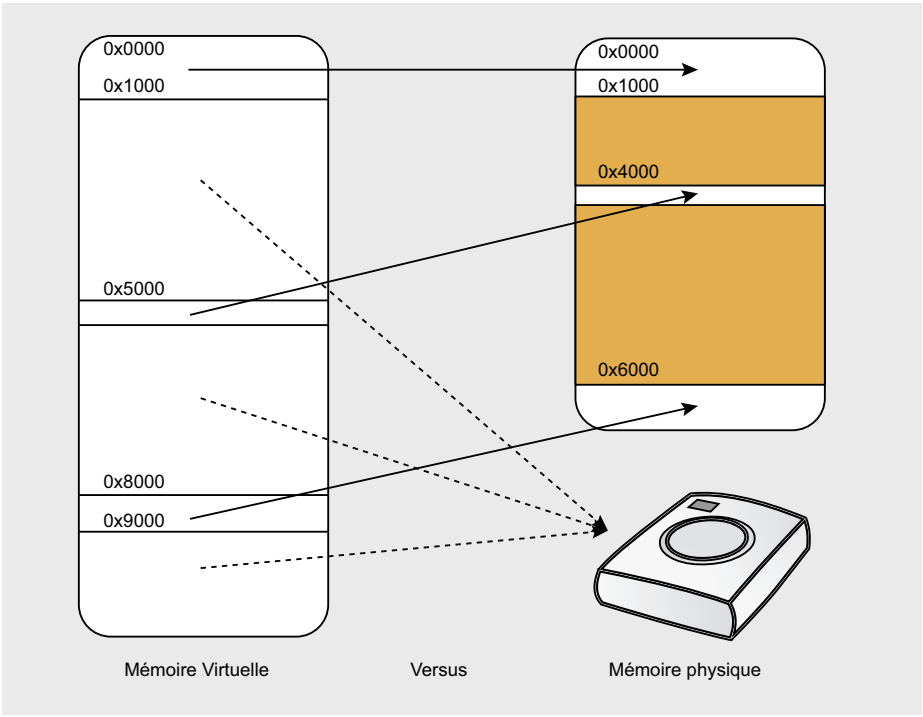

Figure 3. *Représentation de la mémoire utilisée par un programme*

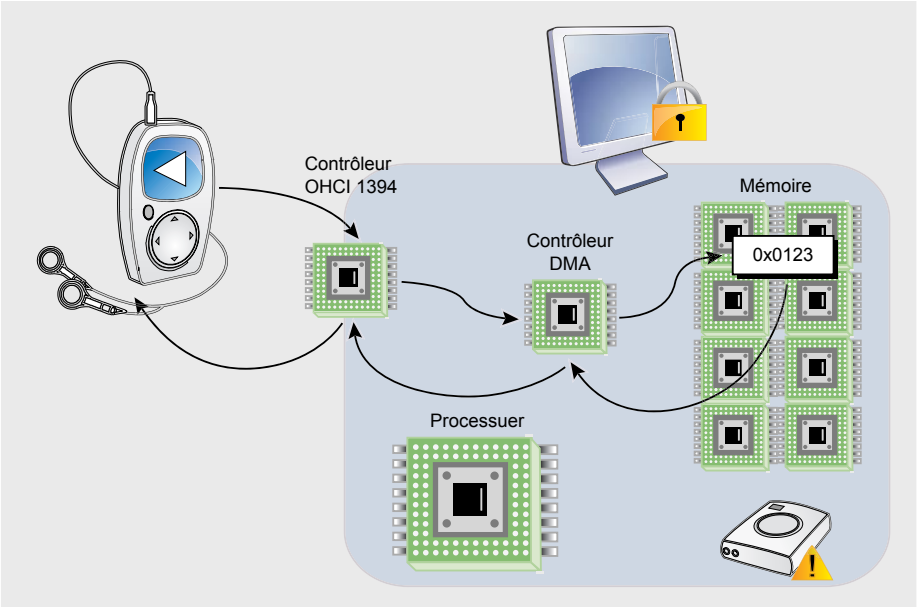

Figure 4. *Connexion par FireWire*

informations sont lisibles après 5 secondes puis 80% après 3 minutes. Comment peut-on récupérer alors ces informations ?

Défi technique

Pour pouvoir réaliser un dump physique de la mémoire d'une machine, il est nécessaire que plusieurs défis techniques soient résolus. Premièrement, le refroidissement de la mémoire. Inutile de chercher de l'azote liquide, il est possible de la refroidir efficacement et à peu de frais grâce aux sprays propulsant de l'air, utilisé entre autres pour éliminer la poussière à l'intérieur des ordinateurs. En effet, la réaction utilisée est endothermique et appliqué pendant quelques secondes à faible distance, ce spray est capable de refroidir des composants dans la gamme de température qui nous intéresse.

Deuxièmement, l'accès physique à la mémoire. Le problème est technique et a plusieurs solutions. La première consiste en la lecture physique sur le même ordinateur. En effet, il n'y a aucune mise à zéro de la mémoire lorsqu'un PC redémarre. Ce comportement est normal et est d'ailleurs repris dans les machines virtuelles comme VMWare. Malheureusement, les ordinateurs portables, principales cibles de cet article, ne possèdent pas de bouton reset. Que ce passe-t-il de plus si un mot de passe a été défini dans le BIOS ? Il existe une deuxième possibilité : si le mode *Quick Boot* est activé, la mémoire n'est pas initialisée au démarrage de la machine. Il suffit d'éteindre et de rallumer la machine pour y avoir accès ou bien de déplacer la mémoire dans un autre ordinateur après l'avoir refroidie. Une dernière solution existe : les supports de stockage basés sur de la mémoire vive. Cette solution n'utilisant pas le format de mémoire qu'on trouve dans les portables, elle ne sera pas abordée dans cet article.

Troisièmement, le dump en lui-même. En effet, il est possible d'accéder à la mémoire vive sous Linux via le périphérique */dev/mem*. Mais démarrer via un livecd par exemple efface plus de la moitié de la mémoire vive. Deux solutions ont été trouvées : démarrer via PXE (par le réseau) pour charger un petit programme qui dump la mémoire via le réseau ou

Terminologie

- *DMA* : L'accès direct à la mémoire ou DMA (acronyme anglais de *Direct Memory Access*) est un procédé informatique où des données circulant de ou vers un périphérique (port de communication, disque dur) sont transférées directement par un contrôleur adapté vers la mémoire principale de la machine, sans intervention du microprocesseur si ce n'est pour initier et conclure le transfert. La conclusion du transfert ou la disponibilité du périphérique peuvent être signalés par interruption.
- *TPM* : *Trusted Platform Module* (également nommé puce TPM ou puce Fritz) est un composant cryptographique matériel, sur lequel s'appuie l'implémentation au niveau matériel du système NGSCB. Il est appelé à être intégré sur les cartes mères des ordinateurs et autres équipements électroniques et informatiques conformes aux spécifications du *Trusted Computing Group*.

lancer un mini-programme via une clé usb qui stocke la mémoire vive sur celle-ci.

Mise en pratique

Si les chercheurs de Princeton n'ont pas rendu leurs utilitaires publics, un chercheur en sécurité a mis à disposition un outil pour le faire. Cet outil s'appelle *msramdump* et est disponible à l'adresse *http://mcgrewsecurity.com/projects/msramdmp/*. On peut se référer pour son installation au Listing 1. Un dump se réalise de la manière suivante. L'outil proposé au Listing 2 écrit en mémoire une très grande chaîne de caractères qui est utilisée pour la démonstration. On exécute cet outil, puis au bout de quelques dizaines de secondes, on éteint l'ordinateur via un appui prolongé sur la touche power puis on rallume aussitôt le PC. Enfin on exécute l'outil *msramdump* grâce à un démarrage sur le périphérique spécialement préparé. Une recherche de la chaîne précédemment créée peut alors être exécutée sur le fichier produit via l'utilitaire *strings*. Dans notre exemple, la commande est `strings /dev/sda2 | grep HACKIN9`.

Concrètement, cet outil est exécuté comme un plugin dans l'outil syslinux (utilisé pour booter sur usb ou sur cd) au plus près de la séquence de démarrage. La mémoire utilisée n'est alors que de 2 Mo dans cet exemple. Ainsi, sur une mémoire de 1024 Mo (1Go) jusqu'à 1022 Mo peuvent être récupérés. On notera que le programme msramdump n'est pas optimisé et nécessite presque 30 minutes pour copier 1Go de mémoire. Cette manipulation est illustrée à la Figure 2.

Première analyse

S'il est possible de récupérer la mémoire vive d'une machine et donc potentiellement en découvrir les mots de passe et les clés de cryptage, il reste néanmoins deux problèmes à résoudre. Premièrement, la corruption des données. En effet, on rappelle que la rémanence n'est pas la même sur tous les bits et donc que des erreurs sont potentiellement introduites. Deuxièmement, les données ont été extraites de manière brute directement depuis les puces mémoires. Or dans tous les systèmes modernes, la mémoire est paginée, c'est-à-dire que les adresses physiques sont masquées par des adresses virtuelles, comme illustré à la Figure 3. Impossible alors de savoir à quelle adresse chercher ces informations. La seule façon possible est d'analyser la mémoire. Alors si on doit analyser toute la mémoire et si le dump n'est pas parfait, peut-on encore récupérer ces informations ? En effet, si une mémoire correspond à 1Go, et si on essayait toutes les combinaisons d'octets stockées en mémoire, il y aurait à peu près 2^{30} clés possibles. C'est certes moins que les 2^{256} combinaisons possibles d'une clé de 256 bits couramment utilisée aujourd'hui, mais à raison d'un million de clés testées par seconde, cela représente tout de même 2^{24} secondes soit presque 200 jours ! À condition en plus que la clé n'ait pas été corrompue.

Pourquoi se limiter à la lecture quand on peut écrire ?

Si l'attaque décrite précédemment est relativement difficile à mettre en place et les programmes non rendus publics, une autre attaque beaucoup plus facile existe. Celle-ci permet notamment d'accéder au contenu de la mémoire par le port FireWire sans que le système en soit averti.

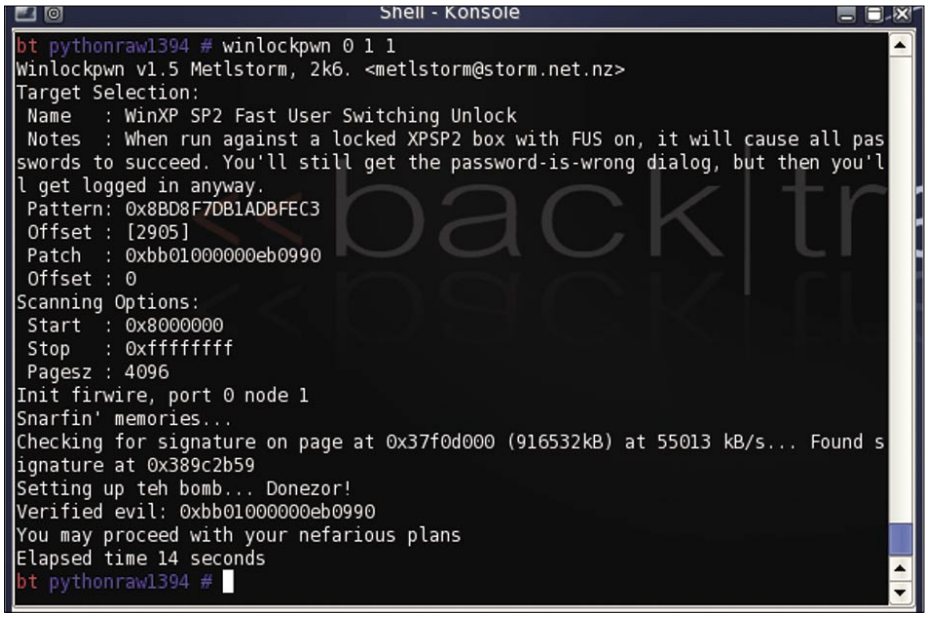

Figure 6. *Contournement du login Windows*

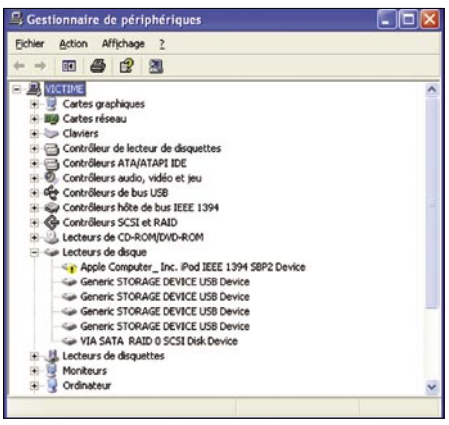

Figure 5. *Détection par Windows du faux ipod*

En effet le FireWire, développé dans les années 90 par Apple, TI et Sony, est un bus de communication série présent sur la grande majorité des ordinateurs portables. À l'inverse de l'usb, ce bus peut utiliser le mécanisme DMA (*direct memory access*) pour transmettre les données pour des applications nécessitant beaucoup de bande passante comme les caméscopes ou les disques durs. C'est un bus où chaque adresse est composée de 48 bits : 16 bits pour le numéro de nœud (périphérique) et 32 bits pour une adresse mémoire. On l'aura compris, ces 32 bits correspondent à une adresse mémoire physique, qu'il est possible de lire ou d'écrire. La Figure 4 illustre cette possibilité.

Cependant, accéder à la mémoire n'est pas si simple car il faut convaincre le système afin d'y avoir accès.

Mise en pratique

L'attaque se réalise de la manière suivante : premièrement l'identifiant de connexion d'un ipod est envoyé, faisant croire au système qu'un appareil vient d'être connecté et qu'il requiert l'utilisation du DMA. Deuxièmement, un canal de communication est établi. Malheureusement, cette attaque est à l'heure actuelle peu documentée et doit être modifiée pour fonctionner. Voici la procédure : Les librairies `libraw1394-dev`, `pythonraw1394` et le logiciel swig sont installés. La librairie `pythonraw1394` permet d'utiliser le *FireWire* depuis le langage python et est téléchargeable depuis le site *http://storm.net.nz/projects/16*. Pour la compiler, le fichier */usr/local/include/raw1394.h* doit être modifié afin de supprimer les `__attribute__ ((deprecated))`. Après remplacement dans le fichier `makefile` de Python2.3 par la version de python installée, il suffit d'exécuter la commande `make` pour compléter l'installation. Le driver du FireWire est chargé via la commande `modprobe raw1394`.

Avant de brancher un câble FireWire entre la machine attaquée et celle de l'attaquant, l'exécution de la commande *romtool -s 0 ipod.csr* est nécessaire. En effet, elle modifie les caractéristiques de l'attaquant de manière à ressembler à un ipod, libre d'accéder à la mémoire. La commande *businfo* permet de vérifier que tout fonctionne et d'identifier les numéros de port et de nœud de la cible une fois connectée. La commande `1394memimage 0 1 mydump -512M` permet alors de réaliser un dump des 512Mo de la machine visée, située sur le port 0 et ayant le numéro de nœud 1. La Figure 5 illustre l'installation sur l'ordinateur cible du faux iPod.

Lors de mes essais, il arrivait que la lecture ne soit plus possible. Une désinstallation du driver du faux iPod depuis Windows a permis de régler le problème.

On notera également que certaines zones mémoires ne sont pas exportées. En effet la lecture non contrôlée de ses zones, comme celles de la mémoire vidéo, peut provoquer le crash du système. Enfin un dump réalisé en dehors de la plage d'adressage réel (en disant par exemple que le système possède 2Go alors qu'il n'en a que 1Go) peut lui aussi conduire au crash du système avec certains chipsets et si celui-ci ne plante pas, les valeurs de ces zones mémoires seront à `0xFF`.

Contourner l'écran de login

Si on vient de montrer qu'il est possible pour un attaquant de lire la mémoire, il lui reste à exploiter la possibilité d'écrire dans celle-ci. L'exploit *winlockpwn* qui a été écrit par l'auteur de la librairie `pythonraw1394` permet de contourner directement par le FireWire l'écran de login Windows. La commande `winlockpwn 0 1 1`, où le numéro de port et de nœud sont ici 0 et 1, permet de contourner l'écran de *Fast User Switching* : tous les mots de passe sont alors acceptés. Cette attaque réalisée en 14 secondes est illustrée en Figure 6. Comment cet outil procède-t-il ? Tout simplement en cherchant une chaîne assez longue caractéristique de *msgina.dll* pour localiser cette dll en mémoire puis en la patchant. Quatre options sont disponibles. Les deux premières pour

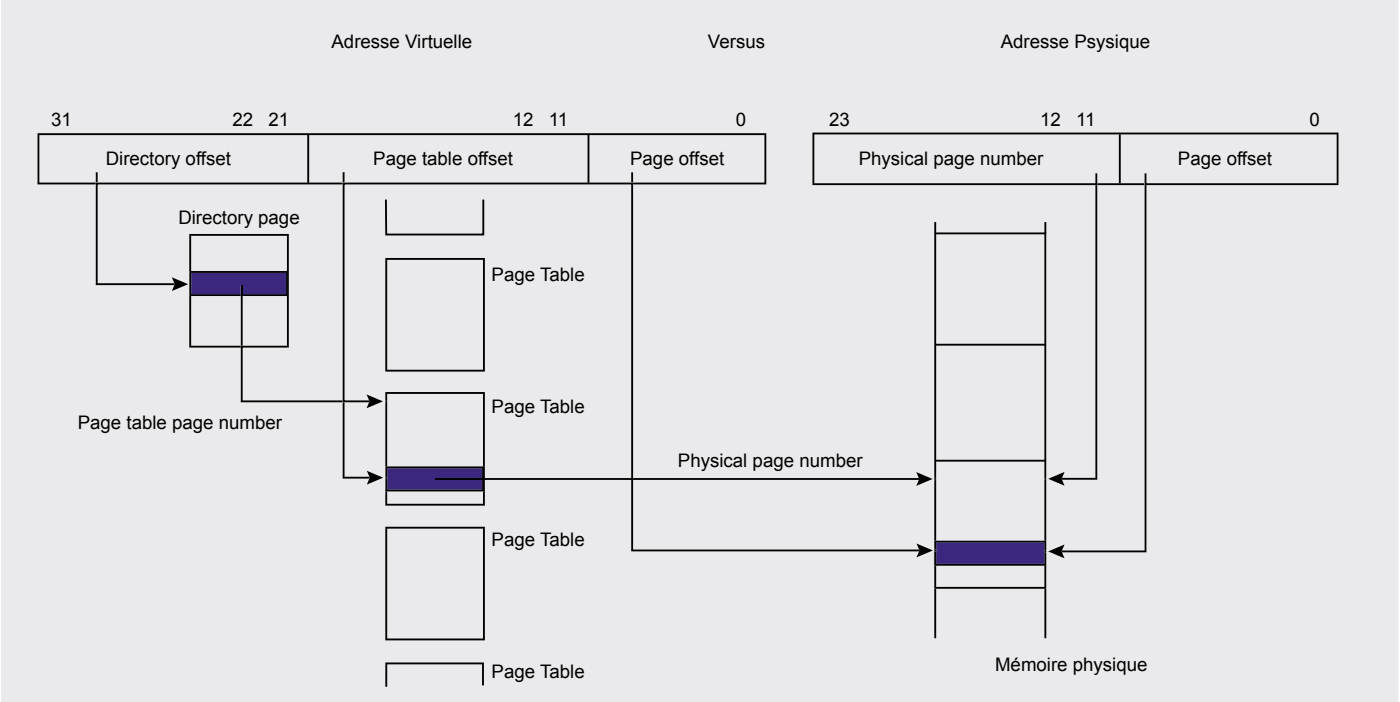

Figure 7. *Adresse virtuelle versus adresse physique*

patcher *msgina.dll* afin que tous les mots de passe soient acceptés, que ce soit dans le *Fast User Switching* ou dans l'écran traditionnel, la troisième pour patcher *msv1_0.dll* afin que le système croit que les comptes n'ont plus de mots de passe et enfin la quatrième pour faire apparaître un *shell* administrateur au milieu de l'écran de login.

Analyse des dumps

Si cette attaque permet de prendre la main sur une machine et donc d'exécuter du code localement, nous n'avons toujours pas analysé le dump d'une machine. En effet, ces dumps produits via FireWire, via une attaque physique ou via un programme exécuté sur la machine elle-même permettent d'obtenir beaucoup d'informations.

Il reste cependant un problème. La mémoire a été copiée de manière brute et il est difficile de savoir où chercher l'information. En effet, la mémoire n'est pas utilisée avec son adresse physique, mais avec son adresse virtuelle, comme expliqué à la Figure 7.

Pour retrouver des informations, la façon la plus basique de procéder consiste à rechercher toutes les chaînes de caractères et à les analyser manuellement. La commande `strings mydump | grep login` permet par exemple de retrouver les mots de passe mémorisés par les navigateurs ou celui de *msn*.

Certaines zones mémoires toujours situées à la même adresse contiennent des données intéressantes, comme le mot de passe du bios. En effet, l'outil *Bioskbsnarf* permet de le récupérer, car il est stocké dans le tampon de l'interruption du mode réel utilisé pour la lecture du clavier après que l'utilisateur l'ait saisi. La Figure 8 illustre cette attaque et montre le mot de passe *password* intercepté. Attention aux scancodes qui ne correspondent pas aux mêmes touches d'un clavier à l'autre et qui sont générés lors de l'appui ou du relâchement d'une touche. Dans ce cas particulier, le mot de passe affiché est *.qsszord*.

Pour une analyse plus poussée de ces dumps, il est tout d'abord nécessaire de connaître le système depuis lequel a été exportée la mémoire. En effet, les techniques d'adressage de la mémoire sont différentes d'un noyau à l'autre. On peut utiliser pour détecter la version d'un Linux la commande : `strings mydump | grep «Linux version»`. Pour Windows, le script *osdection* permet d'obtenir la version du système et de son noyau.

Pour des analyses plus poussées, il est nécessaire de savoir à quelle zone mémoire correspond quel processus. L'outil *PTFinder* permet dans le cadre de dumps réalisés à partir de Windows XP SP2 de connaître la liste des threads et processus ainsi que les zones mémoires concernées qui s'exécutaient au moment de la capture. Il devient alors possible de cibler les recherches dans une zone mémoire précise. La Figure 9 illustre cet outil. On remarquera qu'il est possible de récupérer des informations sur les processus déjà terminés ou de détecter des processus cachés, caractéristiques des rootkits. En effet, l'information liée à ces processus est effacée du noyau lors de leur fin mais pas la zone mémoire dans laquelle ils étaient exécutés. Il est aussi possible d'exporter ces données

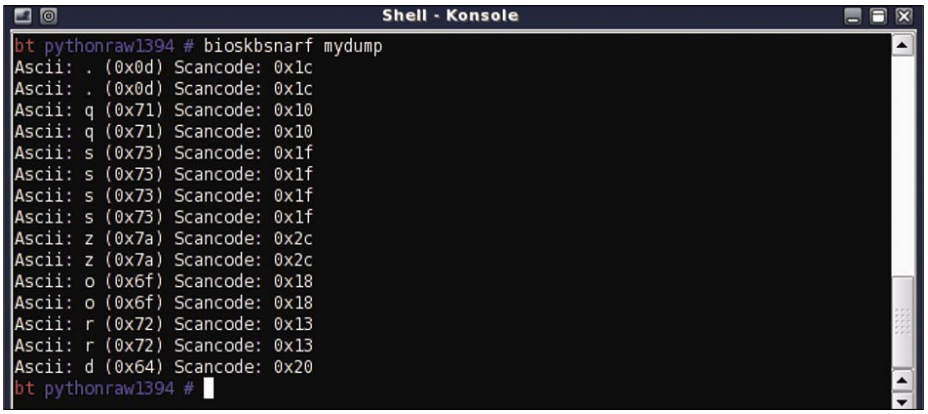

Figure 8. *Récupération du mot de passe du BIOS*

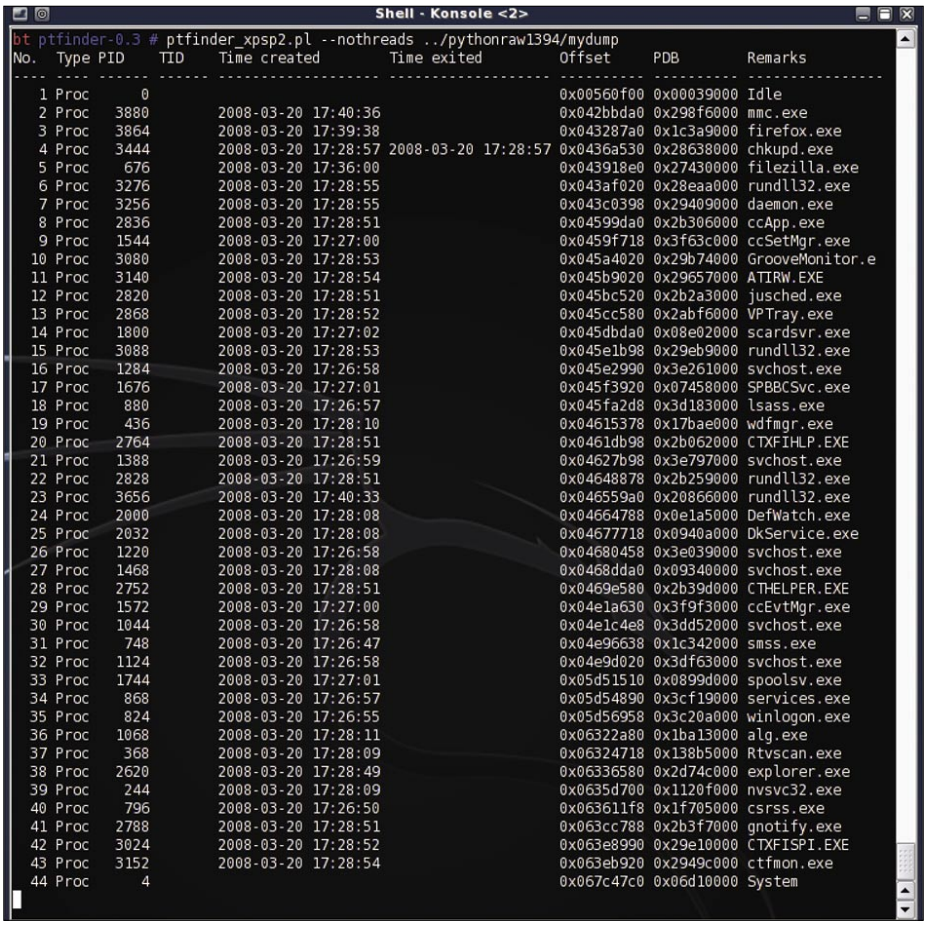

Figure 9. *Liste des processus*

dans le format de dump de Windows, qui est spécifique à cette plateforme, pour analyser le code dans un debuger. L'outil *Volatility* permet quant à lui de récupérer la liste des fichiers et même la liste des sockets ouverts. Bref, il est possible de récupérer de grandes quantités d'information sur l'état du système.

Et le cryptage des disques ?

Comme nous venons de le voir, il existe de nombreuses techniques permettant d'obtenir l'état où se trouvait l'ordinateur au moment où a été réalisée la capture. Jusque-là, le cryptage éventuel mis en place résiste bien puisque seules les données en mémoire sont capturables. Est-il possible de contourner cette protection ? La réponse est oui. Tout d'abord pour la démonstration, on pourra créer un volume crypté et se référer au Listing 3 pour sa mise en place. Il existe 2 manières de procéder au cryptage de fichiers sous Linux : *cryptoloop* ou *dm-crypt*. Si la première peut être considérée comme un bidouillage, la seconde est désormais un standard. C'est cette dernière que nous présentons ici.
Il y a deux façons de mettre en place dm-crypt. Via `dmsetup` ou via `cryptosetup` qui s'appuie sur dm-crypt et qui ajoute un entête LUKS supplémentaire pour faciliter l'utilisation de volumes cryptés. Comme cryptosetup s'appuie sur dm-crypt, nous ne présentons que les attaques spécifiques à dm-crypt. Nous l'avons vu, la commande `strings` permet de récupérer les commandes précédemment rentrées dans le système. Le cryptage ne fait pas exception : si le montage a été exécuté manuellement, un simple grep permet de récupérer le nom du fichier crypté ainsi que sa clé, comme illustré à la Figure 10. Cette technique n'est cependant pas transposable si l'outil est utilisé en mode graphique ou sur Windows.

Listing 3. *Création d'un volume crypté*

```
# obtention d'une clé aléatoire
dd if=/dev/random bs=32 count=1 | hexdump -e '32/1 «%02x» «\n»'
# création d'un fichier de 10Mo
dd if=/dev/zero of=/mnt/sda2/secretfile bs=512 count=20480
losetup /dev/loop0 /mnt/sda2/secretfile
# création et montage du disque crypté
echo «0 'blockdev --getsize /dev/loop0' linear /dev/loop0 crypt aes-plain 95ee94baf656
      14e76bbe195c04ad3f224e4d876a671dff0dde492aca1007590f» | dmsetup create volsecure
mke2fs /dev/mapper/volsecure
mount /dev/mapper/volsecure /mnt/secure
```

Listing 4. *Extrait de /usr/src/linux-2.6.21.5/drivers/md/dm-crypt.c*

```
struct crypt_config {
   struct dm_dev *dev;
   sector_t start;
   /*
      * pool for per bio private data and
      * for encryption buffer pages
   */
   mempool_t *io_pool;
   mempool_t *page_pool;
   struct bio_set *bs;
   /*
      * crypto related data
   */
   struct crypt_iv_operations *iv_gen_ops;
   char *iv_mode;
   union {
      struct crypto_cipher *essiv_tfm;
      int benbi_shift;
   }
   iv_gen_private;
   sector_t iv_offset;
   unsigned int iv_size;
   char cipher[CRYPTO_MAX_ALG_NAME];
   char chainmode[CRYPTO_MAX_ALG_NAME];
   struct crypto_blkcipher *tfm;
   unsigned long flags;
   unsigned int key_size;
   u8 key[0];
};
```

Propriétés des clés

Linux étant un système ouvert, le code source est disponible. Cela permet notamment de connaître avec précision comment est organisé le noyau et en particulier, de quelle manière sont stockées les clés utilisées par dm-crypt. Un extrait du fichier dm-crypt.c est présenté au Listing 4. Comme on peut le voir, la clé est stockée au sein de la structure `crypt _ config`. Attention, cette structure évolue au cours des versions du noyau et on prendra soin pour la suite d'avoir la bonne définition. Elle possède un certain nombre de propriétés : dm-crypt est exécuté dans le noyau et y utilise des zones mémoires. Ses pointeurs sont nuls, soit ils pointent vers des structures situées dans la zone mémoire du noyau, c'est à dire ayant une adresse virtuelle supérieure à `0xc0000000`. On notera que les pointeurs font référence à de la mémoire virtuelle et donc que ceux-ci ne font pas référence à une zone du fichier de dump. Enfin les valeurs de `key-size` et `iv-size` ont de grandes chances de valoir `16` ou `32` qui sont les valeurs standards de longueur de clés.
Le programme *keysearch*, adapté par mes soins pour le noyau `2.6.22.5` et illustré au listing 5, parcourt la mémoire à la recherche de structures répondant à ces critères et les affiche. Il suffit pour adapter le programme à un nouveau noyau de copier-coller la définition de la structure du fichier *dm-crypt.c* dans le fichier *keysearch.c* et de remplacer la définition des pointeurs des structures par des (`void *`). Ceci permet d'éviter d'inclure une de grande quantité de fichiers et donc de limiter les erreurs de compilation. Le programme peut être généré via les commandes : `gcc -c keysearch.c` ; `gcc keysearch.o -o keysearch`. Une fois l'outil exécuté et la clé récupérée, il suffit de monter le volume crypté puis de le parcourir. Un exemple de récupération de clé est donné à la Figure 11.

Discrétion

Deux attaques viennent d'être présentées : la première requiert un démontage en règle du PC visé et la seconde de relier celui-ci à un second PC. Existe-t-il une manière d'agir en toute discrétion avec des grandes chances de réussite ?
La réponse est iPod. En effet, nous avons vu que la deuxième attaque est bien au point, que les ports FireWire sont sur tous les PC portables (au besoin il suffit de brancher une carte FireWire PCMCIA ou Express Port et d'attendre que le système installe automatiquement les drivers) et qu'il suffit d'un PC sous Linux ayant un port FireWire et de 2 minutes. Si un PC portable est relativement voyant, l'iPod est une machine très discrète. En effet, le projet iPod linux permet d'exécuter Linux sur tous les iPods excepté les 2ème et 3ème génération d'iPod nano (qui sont sans support du FireWire) et la 6ème génération d'iPod classique. Ainsi après transformation d'un vieil iPod en iPod linux, celui-ci permet de réaliser un dump mémoire pour une analyse ultérieure ou de patcher l'écran de verrouillage en moins d'une minute.

Comment peut-on se protéger ?

Il n'y a pas de protection connue contre la première faille divulguée ici car elle est architecturale. Cependant, il a été remarqué que les données résidant un certain temps à une même adresse mémoire avaient plus de chance d'être récupérées sans erreur. C'est bien sûr le cas des clés qui sont chargées tout le temps que l'ordinateur est allumé. Il faut alors modifier tous ces outils afin qu'ils déplacent la clé constamment en mémoire, sans avoir la garantie que celle-ci ne pourra pas être récupérée.

Une bonne nouvelle pour les serveurs : ceux-ci sont équipés de mémoire ECC (à correction d'erreur) qui doit être par conception systématiquement mise à zéro lors du démarrage. Impossible alors de transférer les barrettes de mémoire sur un autre PC.

Listing 5. *Source du programme keysearch*

```
/* © Torbjörn Pettersson 2007*/
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h>
typedef unsigned long sector_t;
typedef unsigned char u8;
#define CRYPTO_MAX_ALG_NAME 64
struct crypt_config {
   void *dev;
   sector_t start;
   void *io_pool;
   void *page_pool;
   void *bs;
   void *iv_gen_ops;
   void *iv_mode;
   void *iv_gen_private;
   sector_t iv_offset;
   unsigned int iv_size;
   char cipher[CRYPTO_MAX_ALG_NAME];
   char chainmode[CRYPTO_MAX_ALG_NAME];
   void *tfm;
   unsigned long flags;
   unsigned int key_size;
   u8 key[0];
};
int keysearch(char *mem, int size) {
   int i,j;
   struct crypt_config *cr;
   for(i = 0; i < (size - sizeof(struct crypt_config)); i++,
         mem++) {
      cr = (struct crypt_config *) mem;
      if(
         (void *) cr->io_pool > (void *) 0xc0000000 && (void *)
               cr->tfm > (void *) 0xc0000000 &&
         ((void *) cr->dev > (void *) 0xc0000000 ||
         (void *) cr->dev == (void *) 0x00000000) &&
         (void *) cr->io_pool > (void *) 0xc0000000 &&
         (void *) cr->page_pool > (void *) 0xc0000000 &&
         (void *) cr->iv_gen_ops > (void *) 0xc0000000 &&
         ((void *) cr->iv_mode > (void *) 0xc0000000 ||
         (void *) cr->iv_mode == (void *) 0x00000000) &&
         ((void *) cr->iv_gen_private > (void *) 0xc0000000 ||
         (void *) cr->iv_gen_private == (void *) 0x00000000) &&
         (cr->key_size == 16 || cr->key_size == 32) &&
         (cr->iv_size == 16 || cr->iv_size == 32)
      ) {
         if((long)cr->start > 0)
            printf("offset: %ld blocks\n", (unsigned long int )
                  cr->start);
         printf("keylenght: %d\n", (cr->key_size * 8));
         printf("key: ");
         for(j = 0; j < cr->key_size; j++)
            printf("%02X",cr->key[j]);
         printf("\n");
         printf("cipher: %s\n",(cr->cipher));
         printf("chainmode: %s\n",(cr->chainmode));
      }
   }
   return(0);
}
int main(int argc, char **argv) {
   int fd;
   char *mem = NULL;
   struct stat st;
   if(argc < 2) {
      printf("Usage: %s [memory dump file]\n",argv[0]);
      exit(-1);
   }
   if(stat(argv[1],&st) == -1) {
      perror("stat()");
      printf("Failed to stat %s\n",argv[1]);
      exit(-1);
   }
   fd = open(argv[1],O_RDONLY);
   if(fd == -1) {
      perror("open()");
      printf("Failed to open %s\n",argv[1]);
      exit(-1);
   }
   mem = mmap(0,(int)st.st_size, PROT_READ, MAP_SHARED, fd, 0);
   if(mem == ((void *) -1)) {
      perror("mmap()");
      exit(-1);
   }
   (void)keysearch(mem,(int)st.st_size);
   return(0);
}
```

Sur Internet

- http://citp.princeton.edu/memory/
- http://mcgrewsecurity.com/projects/msramdmp/
- http://storm.net.nz/projects/16
- http://timlegge.blogspot.com/
- http://sandman.msuiche.net/
- https://www.volatilesystems.com/VolatileWeb/volatility.gsp
- http://computer.forensikblog.de/en/2006/03/dmp_file_structure.html
- http://mcgrewsecurity.com/projects/msramdmp/
- http://computer.forensikblog.de/en/2008/02/acquisition_5_firewire.html
- http://computer.forensikblog.de/en/2007/11/ptfinder_0_3_05.html
- http://www.secuobs.com/news/12032007-firewire.shtml
- http://www.friendsglobal.com/papers/FireWire%20Memory%20Dump%20of%20Windows%20XP.pdf
- http://www.forensickb.com/2007/11/extract-mft-records-from-memory-dump.html
- http://windowsir.blogspot.com/2006/10/ptfinder-front-end.html
- http://events.ccc.de/camp/2007/Fahrplan/attachments/1300-Cryptokey_forensics_A.pdf

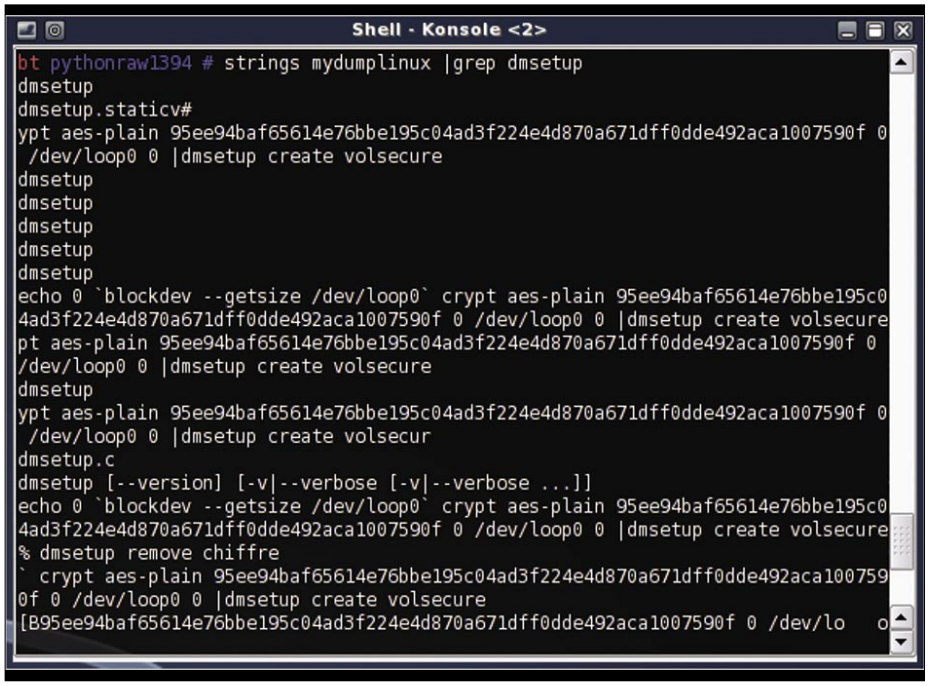

Figure 10. *Recherche des commandes dmsetup*

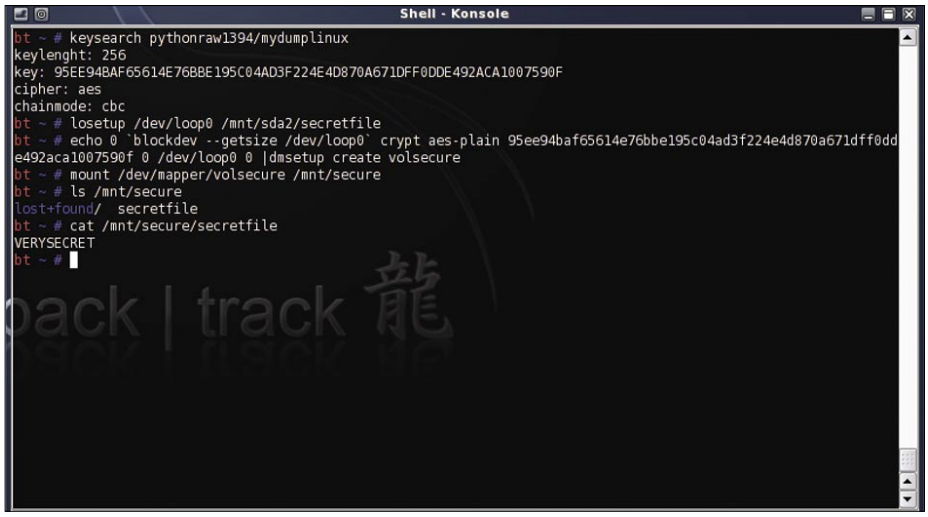

Figure 11. *Recherche de la clé en mémoire puis exploitation*

Les dispositifs de sécurité tels que les cartes à puces ou la puce TPM installés sur la grande majorité des cartes mères sont-ils épargnés par ce type d'attaque ? Pas du tout. Tout d'abord la puce TPM qui stocke la clé de cryptage transmet celle-ci si elle juge le système sûr, ce qui est le cas ici puisque le système est déjà démarré. Les clés privées des cartes à puce éventuellement utilisées ne peuvent pas être récupérées par cette méthode car c'est la carte elle-même qui gère les opérations liées à ses clés mais la clé utilisée pour le cryptage peut être extraite et ainsi décoder le disque. En effet, on rappelle que la cryptographie asymétrique, la plus sûre, utilisée par les dispositifs de cryptage, est extrêmement lente et qu'elle est utilisée uniquement en conjonction avec la cryptographie symétrique pour sécuriser celle-ci.

Concernant le problème d'accès à la mémoire par FireWire, ceci n'est pas un bug mais une fonctionnalité. C'est en effet le comportement souhaité par leur concepteur. Il n'y a donc pas de *patch*. Il est cependant possible de désactiver les ports FireWire (qui en a besoin dans un environnement d'entreprise ?) ou de désactiver le DMA si le driver le permet. Dans tous les cas, il ne sera plus possible d'utiliser le FireWire pour transférer des données. Enfin comme la capture prend du temps, si les données sont constamment déplacées, leur risque d'interception diminue. Si la première attaque est très complexe et comporte un risque d'erreur relativement important, elle est imparable.

Conclusion

Nous venons de le démontrer, il est possible en accédant directement à la mémoire de vaincre les protections d'une station verrouillée : mot de passe du bios, déverrouillage de la session, liste des processus et récupération d'éventuelles clés de cryptage en mémoire, permettant une analyse plus approfondie ensuite. Aucune puce TPM, carte à puce ou lecteur d'empreinte ne peut empêcher de lire la mémoire.

Vincent Le Toux
Après son diplôme de l' ENSIMAG obtenu en 2003, l'auteur a rejoint une société d'études de marché à Paris en tant que chef de projet. Aujourd'hui, il est employé dans l'équipe architecture d'une grande institution financière à Bruxelles.
Pour contacter l'auteur : *vincent.letoux@gmail.com*