

Analyses Forensiques

Dissimulation de données : le disque dur



## Sommaire

Introduction.....	3
1. Les zones constructeurs (HPA/DCO).....	4
1.a La « Host Protected Area ».....	5
1.b La « Device Configuration Overlay ».....	6
2. A la conquête de l'« espace ».....	8
2.a Les slack space.....	8
2.b Les ADS : « Alternate Data Streams ».....	11
Conclusion.....	13

## Introduction

Les disques durs, objet d'analyse de prédilection lors d'une fouille post-mortem, n'ont cessé d'augmenter en capacité de stockage ces derniers temps, ce qui n'est pas sans causer quelques soucis aux investigateurs et experts informatiques.

En effet, le coût d'une analyse forensique étant élevé, l'investigateur dispose généralement de peu de temps pour analyser une quantité monstrueuse de données. Les criminels le savent, et pour empêcher l'investigateur d'arriver à ses fins, où même pour parfois le ralentir, étant donné qu'il suffit de garder les informations cachées le temps de l'analyse, ils ne manqueront pas d'abuser de techniques visant à dissimuler des données.

En dehors des techniques de dissimulations basiques connues par tous comme le changement de l'extension d'un fichier pour le faire passer pour un autre type, ou encore cacher un fichier par ses attributs (sous Windows) ou en le faisant commencer par un « . » (sous Unix), il existe des techniques vraiment efficaces, abusant par exemple des systèmes de fichier de part leur architectures et le détournement de fonctionnalités qu'ils offrent.

Cet article ne se veut en aucun cas être exhaustif, il a pour but de faire découvrir au lecteur des techniques employées dans la vie de tous les jours pour rendre la vie de l'investigateur forensique plus difficile.

## 1. Les zones constructeurs (HPA/DCO)

La première étape lors d'une investigation numérique post mortem (à froid), est la duplication du/des disque(s) dur à analyser. Cette étape vise à fournir à l'enquêteur une copie sur laquelle il pourra travailler. Afin d'avoir une copie parfaite, il est évident qu'il ne faut pas modifier le média lors de la copie, mais également ne rien oublier lors de la duplication, en effet l'oubli d'une zone pourrait laisser passer des informations déterminantes lors de la phase d'enquête.

Parmi ces zones à ne pas oublier, on en retrouve deux qui sont généralement utilisées par les constructeurs. Ces zones sont appelés **HPA** (Host Protected Area) et **DCO** (Device Configuration Overlay).

La HPA est souvent utilisée afin d'y stocker des outils de diagnostic et/ou de restauration système (afin d'économiser les dépenses occasionnées par les dvd de restauration fournis précédemment par les constructeurs), on la retrouve également parfois dans les systèmes « anti-vol » puisqu'elle résiste aux formatages.

La DCO est par contre utilisée afin de permettre aux constructeurs d'acheter différentes marques de disque dur et de choisir la taille sous laquelle apparaîtra le disque dur (par exemple un disque dur de 160Go apparaîtra sous la forme d'un disque dur de 120Go) afin de les faire paraître uniforme.

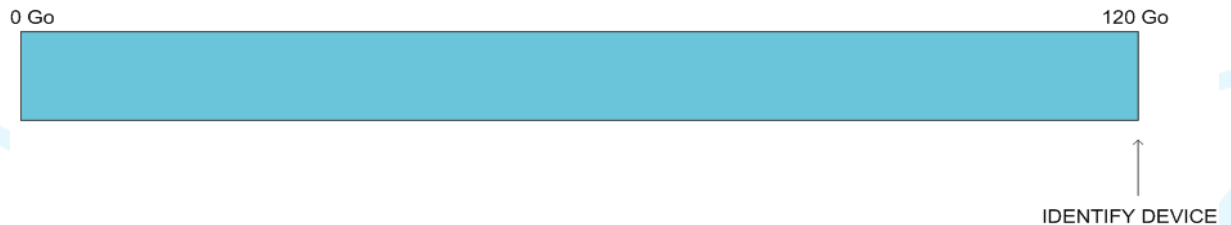
Une des caractéristiques de ces zones qui nous intéresse ici, est qu'elles ne sont normalement pas visibles depuis un système d'exploitation ou le BIOS, ce qui rend leur manipulation/détection non aisée par les utilisateurs.

Ces deux zones sont toutefois modifiables via l'utilisation d'outils spécialisés, ce qui permet à des individus ayant connaissance de leur existence de pouvoir y cacher des informations. Cette dissimulation peut même fonctionner vis-à-vis de certains outils servant lors d'une analyse forensique puisqu'il arrive que ceux ci ne prennent pas en charge ces zones.

La gestion de ces zones s'effectue par l'intermédiaire de registre gérés par le contrôleur de disque, ces registres stockent différents types de valeurs qui sont consultables et modifiables par l'intermédiaire de commandes ATA comme « IDENTIFY DEVICE » qui récupère le numéro du dernier secteur accessible pour l'utilisateur sur le disque.

## 1.a La « Host Protected Area »

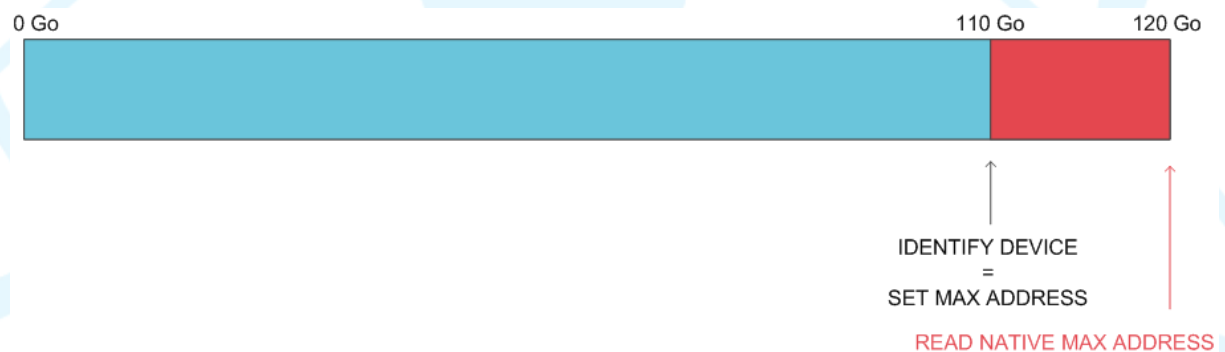
Prenons tout d'abord le cas d'un disque dur d'une capacité de 120Go, n'ayant aucune zone protégée. Dans ce cas ci, l'ensemble du disque dur est accessible, ainsi le nombre total de secteurs accessibles par l'utilisateur correspond au nombre total de secteurs composants le disque dur :



Prenons maintenant le cas où un utilisateur souhaite créer une zone HPA afin de pouvoir y dissimuler des données. Pour ce faire il va devoir passer par l'intermédiaire de la commande ATA « SET MAX ADDRESS » qui va lui permettre de définir le numéro du dernier secteur accessible à l'utilisateur, autrement dit il va redéfinir la taille de la zone accessible à l'utilisateur.

Admettons qu'il souhaite se réserver 10Go, il va placer les 110 premiers Go comme étant lisibles, ainsi la zone s'étendant des 110 aux 120 Go sera considérée comme une zone protégée puisque non accessible.

Le seul moyen de détecter la présence d'une HPA est de faire appel à la commande ATA « READ NATIVE MAX ADDRESS » qui retourne le numéro du dernier secteur auquel une commande ATA peut accéder, ainsi si la valeur retournée est différente de celle retournée par « IDENTIFY DEVICE » c'est qu'il existe une zone HPA.



Afin de pouvoir manipuler cette zone, il faut remplacer la valeur retournée par « IDENTIFY DEVICE » par celle retournée par « READ NATIVE MAX ADDRESS » grâce à la commande « SET MAX ADDRESS ». Ainsi la zone fait à nouveau partie des secteurs manipulables par les utilisateurs, nous pouvons donc y stocker nos données puis redéfinir la HPA comme précédemment avec la commande « SET MAX ADDRESS ».





## 2. A la conquête de l'« espace »

Les données sont stockées et organisées sur un disque dur par l'intermédiaire d'un système de fichier, celui-ci permettant la mise en place d'une structure à travers laquelle l'utilisateur perçoit les données.

Il existe des spécifications qui expliquent le fonctionnement de ces systèmes de fichiers, et comme n'importe quel format de fichier, l'étude de ces spécifications permet de déduire des endroits dans lesquels nous pourrions y dissimuler quelque chose.

Mais en dehors des « faiblesses » dans l'architecture du système de fichier, il existe parfois des « fonctionnalités », implémentées pour des raisons de compatibilité, qui permettent la dissimulation de données de façon transparente pour l'utilisateur comme nous le verrons plus loin.

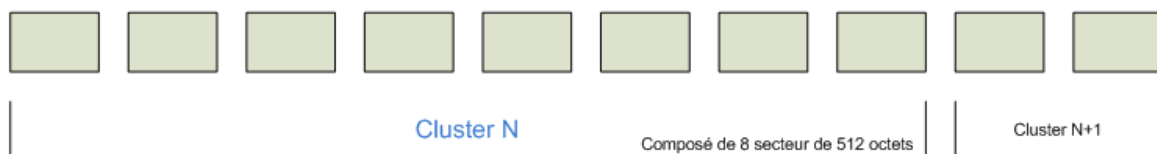
Cette partie donne un aperçu des techniques utilisées et ne se veut pas exhaustive.

### 2.a Les slack space

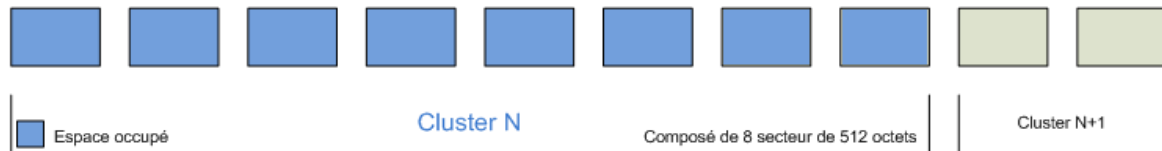
Lors de la création d'un fichier sur le disque dur, celui-ci alloue un ou plusieurs clusters dédiés au fichier en fonction de sa taille.

Un cluster est la plus petite unité allouable sur un disque dur, il est en fait composé d'un groupe de secteurs (ce nombre dépend de la taille de la partition et du système de fichier utilisé).

Nous prendrons ici le cas d'un disque dur dont les secteurs ont une taille de 512 octets, et dont le cluster fait 4096 octets ( $4096/512 = 8$ , soit 8 secteurs par cluster).



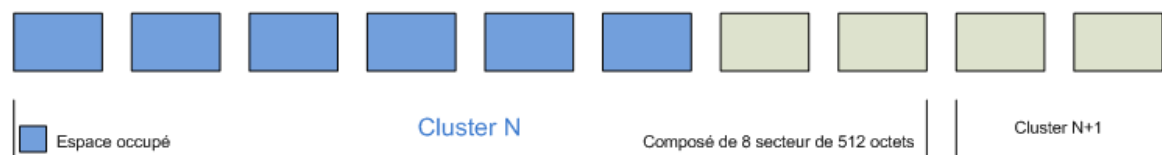
Ainsi notre système alloue un nombre multiple de 4ko pour chaque fichier que l'on stocke, ce qui, comme vous l'avez peut être deviné apporte quelques inconvénients. Tout d'abord, dans le cas où le fichier a une taille multiple de 4096, son contenu rentrera parfaitement dans le(s) cluster(s) alloué(s). Rien ici ne pose problème.



Dans le cas contraire, où notre fichier à une taille inférieure à un nombre multiple de 4096, comme dans le schéma ci-dessous, celui-ci n'occupera pas tout l'espace qui lui est dédié. Que contient donc l'espace qu'il reste ?

Celui-ci contiendra bien souvent les données qui occupaient précédemment cet espace et qui n'ont depuis pas été réécrites.

Cela peut donc servir lors de la récupération de données effacées, mais aussi dans la dissimulation de données, puisqu'il nous suffit de placer, à partir de la fin du fichier stocké dans le cluster jusqu'à la fin du cluster, des données que nous voulons cacher.



Pour bien comprendre le principe, on pourrait par exemple prendre un système d'archivage où l'on attribue des casiers en fonctions des domaines. Un casier serait dédié à un domaine x, mais le domaine x pourrait très bien être constitué seulement de quelques dossiers, ces dossiers ne rempliraient pas l'espace offert par le casier en terme de profondeur, l'espace qu'il reste est le *slack space* dont nous parlions plus haut.

Voici un exemple de l'exploitation de cette technique par l'utilisation de l'outil **bmap** sous linux:

```
sh4ka@mini-moi:~$ stat /etc/passwd
File: `/etc/passwd'
Size: 1623      Blocks: 8      IO Block: 4096  fichier régulier
Device: 801h/2049d  Inode: 3402101  Links: 1
Access: (0644/-rw-r--r--) Uid: (  0/   root)  Gid: (  0/   root)
Access: 2009-08-02 23:04:41.000000000 +0200
Modify: 2009-06-25 08:40:06.000000000 +0200
Change: 2009-06-25 08:40:06.000000000 +0200
sh4ka@mini-moi:~$ md5sum /etc/passwd
6d7017c1fd892e4b5a33989d33803df1  /etc/passwd
sh4ka@mini-moi:~$ cat secret
Ceci est un exemple de message secret!!
sh4ka@mini-moi:~$ sudo bmap --mode slack /etc/passwd
getting from block 13599420
file size was: 1623
slack size: 2473
block size: 4096
sh4ka@mini-moi:~$ cat secret |sudo bmap --mode putslack /etc/passwd
stuffing block 13599420
file size was: 1623
slack size: 2473
block size: 4096
sh4ka@mini-moi:~$ sudo bmap --mode slack /etc/passwd
getting from block 13599420
file size was: 1623
slack size: 2473
block size: 4096
Ceci est un exemple de message secret!!
sh4ka@mini-moi:~$ stat /etc/passwd
File: `/etc/passwd'
Size: 1623      Blocks: 8      IO Block: 4096  fichier régulier
Device: 801h/2049d  Inode: 3402101  Links: 1
Access:(0644/-rw-r--r--) Uid: (  0/   root)  Gid: (  0/   root)
Access: 2009-08-02 23:04:41.000000000 +0200
Modify: 2009-06-25 08:40:06.000000000 +0200
Change: 2009-06-25 08:40:06.000000000 +0200
sh4ka@mini-moi:~$ md5sum /etc/passwd
6d7017c1fd892e4b5a33989d33803df1  /etc/passwd
```

Comme nous pouvons le voir, cette méthode possède plusieurs avantages :

- Elle est simple à mettre en place
- Furtive (peut l'être encore plus en chiffrant les données que l'ont stocke et en les découpant sur plusieurs slack space différents)
- Ne modifie en rien le fichier « hôte » (la somme MD5 et les dates de modification/accès ne sont pas modifiées)

Malgré tout, comme toutes les méthodes, elle possède un inconvénient majeur : si le fichier est modifié, il y a des risques que le slack space soit réduit, ce qui pourra provoquer la corruption des données que l'on souhaite dissimuler. Pour éviter que cela se produise, il est préférable de stocker nos informations dans des fichiers dont le contenu a très peu de chance d'être modifié.

Une variante de la technique du slack space, serait d'utiliser ce que l'on appelle l' « unallocated space » (l'espace non alloué), c'est à dire l'espace qui n'est pas ou plus utilisé par un fichier.

Lors de la suppression d'un fichier, par exemple, l'espace occupé par le fichier est juste marqué comme non alloué, il n'est pas pour autant physiquement effacé. On peut donc écrire des données dans ces espaces, ou les récupérer dans le cadre d'une investigation forensique.

Mais encore une fois le placement de données y est risqué, puisque la zone peut très bien être allouée par le système de fichier, mais ceci peut cependant être « contourné ».

Il est en effet possible d'indiquer les secteurs où l'on cache des données, comme étant défectueux, ainsi le système de fichier n'allouera pas ces secteurs pour y stocker des données, malgré tout certains outils permettent la détection des faux secteurs défectueux, cette technique est donc loin d'être parfaite (comme toutes les autres).

## **2.b Les ADS : « Alternate Data Streams »**

Nous parlons plus haut de fonctions ajoutées pour des raisons de compatibilité, c'est ici le cas des ADS.

Les ADS ou « Alternate Data Streams » pour les intimes, ont été implémentés dans le système de fichier NTFS afin de permettre la compatibilité avec le HFS (Hierarchical FileSystem), un système de fichier utilisé par Mac.

Avec le système de fichier FAT, par exemple, il n'y avait qu'un seul stream, celui des données du fichier.

Dans le système de fichier NTFS c'est différent, il y en existe plusieurs, par exemple, un stream qui définit les propriétés du fichier (comme les droits d'accès), un stream pour le contenu du fichier,... Mais les utilisateurs peuvent également rajouter leurs propres streams (des fichiers) qui seront reliés de façon invisible au fichier hôte, ce sont les ADS.

Les ADS peuvent être rattachés aussi bien aux fichiers qu'aux répertoires, de plus ils peuvent contenir tout type de données, allant d'un simple fichier texte jusqu'à un fichier exécutable.

De plus, rien parmi les outils fournis de base dans Windows ne permet leur détection (exception faite de Windows Vista qui inclut un paramètre permettant le listing des ADS dans la commande « *dir* »), ce qui ne fait aucune différence pour l'utilisateur lambda entre un fichier vierge de tout ADS et un fichier « porteur ».

A noter également que les ADS n'ont pas d'autre limite en taille que celle qu'offre l'espace de stockage du disque dur, ce qui est un avantage comparé aux « slack space ». Autre avantage également : il n'y a aucun risque de réécriture involontaire.

La création des ADS se fait très facilement, l'exemple ci-dessous illustre la création d'un fichier rk.exe comme étant un ADS du dossier C:\Windows\ :

```
C:\> type rk.exe > C:\Windows:rk.exe
```

Pour créer un ADS, il suffit donc d'ajouter au nom d'un répertoire ou d'un fichier un « : » puis le nom de notre flux.

Maintenant nous pouvons le lancer de la façon suivante:

```
C:\> start C:\Windows:rk.exe
```

Tel qu'évoqué précédemment, un ADS peut aussi être contenu dans un simple fichier, mais rien ne change sur le plan manipulation :

```
C:\> md5sum c:\windows\notepad.exe
```

```
\ac58d8a9201d6bd43b8417f5478e3ef1 *c:\\windows\\notepad.exe
```

```
C:\> echo Exemple de texte dans un ADS >
```

```
C:\windows\notepad.exe:secret
```

```
C:\> more <c:\windows\notepad.exe:secret
```

```
Exemple de texte dans un ADS
```

```
C:\> md5sum c:\windows\notepad.exe
```

```
\ac58d8a9201d6bd43b8417f5478e3ef1 *c:\\windows\\notepad.exe
```

Nous pouvons constater que leur création ne nécessite aucun privilège et que leur présence est invisible pour l'utilisateur puisque même le hash MD5 est identique.

Sur le plan de la furtivité, certains anti-virus par exemple, ne prennent pas en charge les ADS lors des scan, ce qui permet la dissimulation de données malicieuses. Leur détection s'effectue par l'intermédiaire d'outils spécialisés comme « Streams » de *Mark Russinovich*.

```
C:\> streams -s C:\windows
```

```
Streams v1.56 - Enumerate alternate NTFS data streams
```

```
Copyright (C) 1999-2007 Mark Russinovich
```

```
Sysinternals - www.sysinternals.com
```

```
C:\windows\NOTEPAD.EXE:
```

```
:secret:$DATA 31
```

## Conclusion

Cet article n'est en aucun cas exhaustif, il n'a pour but que de donner un aperçu des techniques pouvant être employées par des utilisateurs voulant dissimuler des données.

Comme nous l'avons vu, aucune de ces techniques n'est parfaite, mais le but n'est pas de totalement dissimuler des données, mais plutôt de retarder le moment où la personne en charge de l'examen du disque dur tombera dessus ;)

**André Moulu**  
[andre.moulu@sh4ka.fr](mailto:andre.moulu@sh4ka.fr)