Il court, il court, le fichier...

Pierre-Yves Bonnetain
py.bonnetain@ba-consultants.fr

B&A Consultants – BP 70024 – 31330 Grenade-sur-Garonne

31 mars 2011

B&A Consultants

- Cabinet de conseil en sécurité informatique créé en 1996.
- Conseils, suivi et assistance en sécurité informatique.
- Audits de sécurité, de configurations, de code...
- Tests d'intrusion, tests d'applications (boîte blanche, boîte noire)
- Analyses de risques, gestion des risques sur l'information
- Ingéniérie de la sécurité informatique, recherche de solutions
- Formations à la sécurité informatique
- Expertise judiciaire (civile ou pénale) et expertises privées

B&A CONSULTANTS SECURITE INFORMATIQUE

Plan

Plan

Contextes des recherches

Généralement une situation *a posteriori* :

- Détection de copies ou transferts non autorisés
- Recherche d'activités délictueuses (contenu protégé ou illégal)

Se heurte à plusieurs difficultés :

1. le fichier peut être « là »,
   - caché dans un coin du système de fichiers
   - sous un nom différent
   - contenu modifié
2. le fichier a transité sur le disque mais n'y est plus

Principales techniques

Deux grandes familles de techniques :

1. Recherches dans le système de fichiers : `find` et autres, pré-calculs de condensats...
2. Recherches sur le support de stockage, « hors » système de fichiers : « carving », recherches de signatures...

Plan

Examen du système de fichiers

- Suppose que le fichier est présent sur le disque dur
- Et présente une caractéristique particulière (nom, condensat, type...)
- Trouvera le fichier où qu'il soit dans le système de fichiers

En bref

Plus les caractéristiques recherchées sont discriminantes, plus la recherche est affinée.

Inconvénients

Ces « caractéristiques » sont faciles à modifier. Si le fichier a été effacé, il ne sera pas trouvé.

Recherche par nom

Un `find` très classique.

```
# find / -type f \
    -name 'Intégrale Jacques Brel.mp3'
```

Evidemment

Si le nom du fichier a été modifié, la recherche échouera.

Recherche par condensats

- Fichier connu $\Leftrightarrow$ contenu connu
- Pré-calcul de condensats (MD5, SHA1...) des contenus.
- Examen de chaque fichier, calcul et comparaison de son condensat.

A noter

Il existe des bases de fichiers connus (http://www.nsrl.nist.gov/). Facile de s'en fabriquer une.

Inconvénient

Modification mineure contenu $\Rightarrow$ condensat différent

Note

`tripwire`, `Aide` et autres reposent sur un principe similaire (pour détecter des modifications indûes).

Conclusions

Ca marche pas trop mal

- Dans des cas simples.
- Pratique dans des situations de « comparaison d'états » (cliché de la machine virtuelle avant/après une infection)
- Résistant aux renommages de fichiers et aux répertoires cachés ou improbables.
- Possibilités de contournement faciles
- Et surtout, ne voit pas les fantômes (fichier effacé)

$\Rightarrow$ s'affranchir du système de fichiers ?

B&A CONSULTANTS SECURITE INFORMATIQUE

Plan

Extraction de données

Data carving dans la langue de Shakespeare.

- Recherches souvent faites dans l'espace libre d'un système de fichiers
- Peuvent être faites sur des interceptions réseau

Principes sous-jacents :

- le début du fichier (en-tête « discriminant ») existe
- le fichier n'est pas fragmenté (sur le disque)
- le fichier n'est pas compressé ou chiffré.

`photorec` est un bon exemple de programme d'extraction de données.

Fonctionnement général

- Identification de l'en-tête du fichier (JPEG, GIF, MP3...)
- Calcul (si possible) de la taille du fichier
- Extraction des blocs du fichier, souvent considérés comme contigus.

Attention

Pourcentage de faux positifs significatif. Il faut ensuite vérifier chacun des fichiers extraits.

La vérification peut être faite via le calcul de condensats (au moins pour éliminer les doublons dans l'extraction).

Extraction de données – suite

- Permet « seulement » une recherche large (dans et hors du système de fichiers)
- Efficace si fichiers structurés avec informations d'identification (en-tête. . .)
- Nécessite une phase de validation des éléments renvoyés
- Et ils peuvent être nombreux !
- Plus l'effacement des fichiers est ancien, moins on aura de résultats.

A noter

A partir d'un bloc identifié, possible de remonter à l'ensemble des blocs du fichier même s'il est fragmenté. Fichier pas effacé ⇒ aucune difficulté. Fichier effacé ⇒ on peut avoir de la chance.

Extraction de données – conclusions

Opération moyennement satisfaisante

- Permet d'examiner toute la surface de stockage (ou toute la capture réseau)
- Suppose un certain nombre d'invariants, dont notamment des « signatures » discriminantes

Notons quand même

Ca marche, notamment dans des situations où les fichiers ont été copiés puis effacés « il y a peu de temps ».

Plan

Remerciements

DFRWS 2010

M. Simson Garfinkel a fait une présentation de l'outil `frag_find` au Digital Forensic Research WorkShop 2010. Un certain nombre de points ci-après sont inspirés de cette présentation, avec son aimable autorisation.

Plan

Idée de base

Fichier = succession de blocs de stockage (ou de paquets réseau).
Descendre d'un niveau dans l'identification par condensats :

- non plus au niveau du fichier complet (XXX Mo)
- mais à celui du bloc de stockage sur le support (512 octets en général)

Un peu de maths

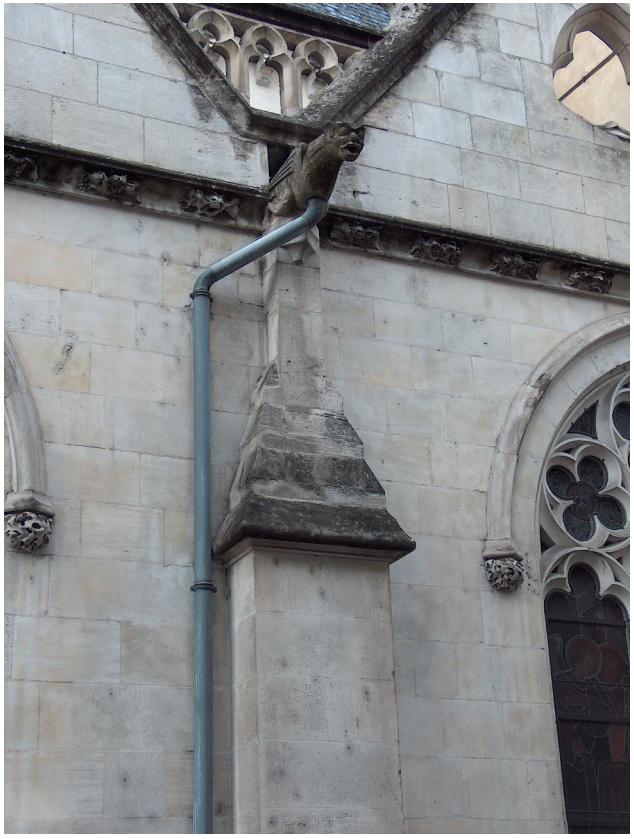
512 octets $\Rightarrow 2^{512\times 8} \approx 10^{1233}$ possibilités différentes.

Notion de « blocs discriminants » : blocs à forte entropie (clés privées, données numériques...).

Hypothèses

1. Si on identifie **un seul** bloc discriminant d'un fichier sur un support, il est probable que l'ensemble du fichier s'y soit trouvé.
2. Si un fichier est créé par un processus à forte entropie, et s'il est prouvé que les blocs de ce fichier sont discriminants par rapport à un corpus connu et étoffé, alors les blocs du fichier sont discriminants.

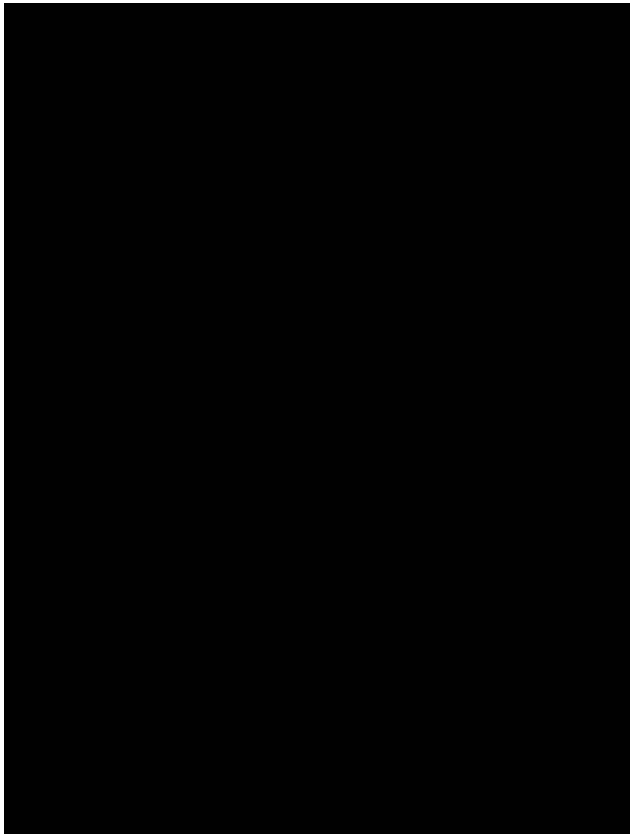
Exemple : image photographique



- Fichier de 183075 octets
- Calcul des condensats MD5 de chaque bloc de 4096 octets

```
$ perl md5blocs.pl Exemple-1.jpg
45 blocs et 183075 octets lus
0 collisions de condensats
```

Contre-exemple



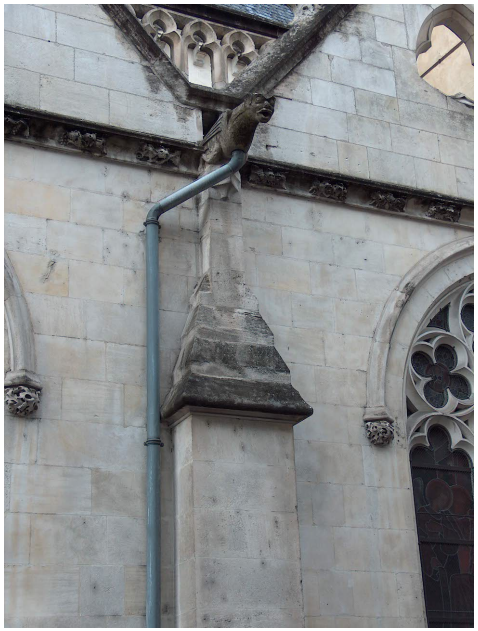
- Fichier de 7695 octets, blocs de 512 octets

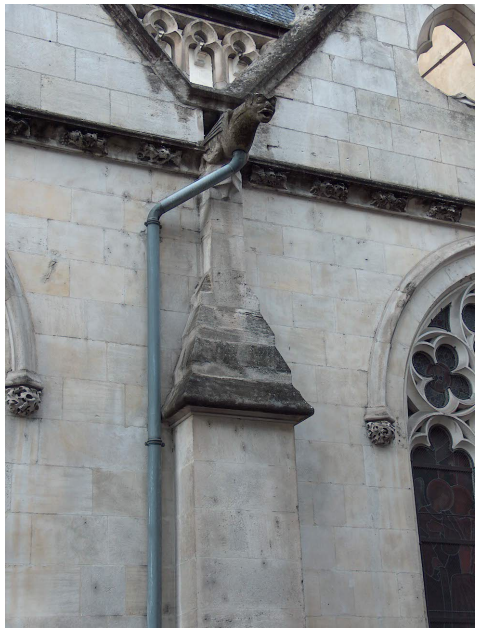
```
$ perl md5blocs.pl Exemple-3.jpg
Bloc 2, MD5 identique à 1
Bloc 3, MD5 identique à 1, 2
Bloc 4, MD5 identique à 1, 2, 3
[...]
Bloc 14, MD5 identique à 1, 2, 3, 4, 5, 6,
  7, 8, 9, 10, 11, 12, 13
Exemple-3.jpg
16 blocs et 7695 octets lus
13 collisions de condensats
```

- Mais est-ce vraiment un fichier à forte entropie ?

B&A CONSULTANTS SECURITE INFORMATIQUE

Exemple 2





Différence entre les deux fichiers peu visible. Résultat : aucun bloc commun, y compris l'en-tête

```
$ perl md5blocs.pl Exemple-1.jpg Exemple-1bis.jpg
Exemple-1.jpg : 45 blocs et 183075 octets lus
0 collisions de condensats
Exemple-1bis.jpg : 45 blocs et 182891 octets lus
0 collisions de condensats
0 collisions inter-fichiers
```

B&A CONSULTANTS SECURITE INFORMATIQUE

Fichiers et blocs discriminants ?

Il existe de nombreux types de fichiers contenant des blocs discriminants :

- vidéos, musique,
- fichiers chiffrés,
- texte avec contenu spécifique,
- fichiers avec « quelques » caractères aléatoires : il existe $10^{33}(\frac{512!}{500!})$ blocs différents contenant 500 zéros et 12 espaces.

Il existe aussi des fichiers sans blocs discriminants :

- contenu constant
- répétition d'une suite constante de caractères (`AB CD EF AB CD EF`)

Sur un support de stockage

- Fichiers alignés sur des frontières de blocs
- Fichiers contenant blocs discriminants ⇒ blocs de stockage discriminants

Conclusion

Trouver un bloc de stockage discriminant est une preuve que le fichier originel a été présent sur le disque dur.

Notez qu'il peut toujours être présent et identifiable !

Plan

L'outil `frag_find`

Développé par Simson Garfinkel, http://afflib.org/.
Principe :

- un ou plusieurs fichiers de référence
- une image disque (ou capture réseau)
- calcul du condensat de chaque bloc de chaque fichier de référence
- examen de l'image disque, calcul du condensat de chaque bloc de stockage
- détection des collisions de condensats.

Note

Bloc du disque produit condensat présent dans plusieurs fichiers référence $\Rightarrow$ affecté au fichier ayant le plus de collisions avec le disque.

Utilisation de `frag_find`

- Détection de fuite de documents sensibles
- Détection de présence, y compris ancienne, de fichiers litigieux.

En temps réel

Difficile sur des accès disques locaux (sauf pilote disque modifié). Sur un accès au travers du réseau, par contre, tout à fait applicable.

Soulignons que

Même si c'est « mieux » lorsqu'on trouve tous les blocs, il suffit d'un bloc discriminant retrouvé pour prouver le transit du fichier sur l'élément analysé.

Méthodologie

- Plutôt que de constituer une base de condensats de contenus complets. . .
- . . . Créer une base de condensats de blocs de données
- Condensat d'un bloc identifié « quelque part » où il ne devrait pas l'être
  - levée d'une alerte (transit réseau)
  - le fichier auquel il appartient est/a été présent
  - ou bloc en question pas discriminant

Intérêt de cette approche

Rapide et efficace Une clé USB de 4 Go peut contenir plusieurs milliards de condensats pré-calculés

Peu de faux positifs Si les blocs non discriminants sont rapidement identifiés

Sûr Inutile de fournir le(s) fichier(s) de référence aux analystes !

Analyse probabiliste des supports à forts volumes

- Disque de 1.5 Tio = 3 milliards de blocs de 512 octets.
- Echantillonage aléatoire du disque, plutôt qu'analyse complète
- Même conclusions ensuite, détection d'un condensat connu = présence du fichier associé (ou bloc non discriminant).

Attention toutefois

Bien calibrer taille échantillon à examiner, par rapport tailles fichiers de référence.

Plan

Faux négatif

Échantillonnage ⇒ risque de « passer à côté » de l'élément probant : faux négatif typique.

Probabilité d'échec

Probabilité de conclure de façon erronée « le fichier n'est pas présent (ou n'a pas transité) sur ce support ».

Intérêt : minimiser cette probabilité, sans pour autant examiner tout le support.

Exemples

- Disque de 1.5 Tio $\rightarrow 3 \times 10^9$ blocs de 512 octets
- Fichier de référence de 100 Mio $\rightarrow 2 \times 10^5$ blocs de stockage

Exemples

- Disque de 1.5 Tio $\rightarrow 3 \times 10^9$ blocs de 512 octets
- Fichier de référence de 100 Mio $\rightarrow 2 \times 10^5$ blocs de stockage
- Echantillon composé d'un seul bloc
  - Probabilité de détection $\frac{2\times10^5}{3\times10^9} = 6.66 \times 10^{-5}$. C'est mieux que le loto, malgré tout...
  - Probabilité d'échec $\frac{3\times10^9-2\times10^5}{3\times10^9} = 0.999933$, soit 99,993%.

Exemples

- Disque de 1.5 Tio $\rightarrow 3 \times 10^9$ blocs de 512 octets
- Fichier de référence de 100 Mio $\rightarrow 2 \times 10^5$ blocs de stockage
- Echantillon composé d'un seul bloc
  - Probabilité de détection $\frac{2\times10^5}{3\times10^9} = 6.66 \times 10^{-5}$. C'est mieux que le loto, malgré tout...
  - Probabilité d'échec $\frac{3\times10^9-2\times10^5}{3\times10^9} = 0.999933$, soit 99,993%.
- Echantillon de n blocs, disque de D blocs, référence de R blocs, probabilité d'échec sur tous les blocs $p = \prod_{i=1}^{n} \frac{D-(i-1)-R}{D-(i-1)}$

Ce qui nous donne

Disque de 1 To

Fichier de référence de 10 Mo, probabilité d'échec en fonction du nombre d'échantillons

Echantillons	Proba échec
1	0.999990
100	0.999000
1000	0.990050
10000	0.904837
100000	0.367868
200000	0.135320
300000	0.049775
400000	0.018308
500000	0.006733

10000 échantillons, probabilité d'échec en fonction de la taille du fichier recherché

Référence	Proba échec
10 Mo	0.904837
50 Mo	0.606522
100 Mo	0.367860
150 Mo	0.223104
200 Mo	0.135308
250 Mo	0.082059
300 Mo	0.049764
400 Mo	0.018301
500 Mo	0.006729

Résultat des courses – en théorie

Recherche sur un disque de 1.5 To (USB2 : 40 Mio/s observé, soit 82000 blocs/s), fichier référent de 100 Mo.

	Disque complet	10000	50000	100000
Durée	595 minutes	$<$ 1 sec	$<$ 1 sec	1.5 sec
Volume lu	1.5 To	5 Mo	25 Mo	50 Mo
Proba faux négatif	0%	36,78%	0,67%	0,0045%

Attention

Les valeurs pour la recherche par échantillonage sont **théoriques**.

Résultat des courses – en pratique

Recherche sur un disque de 100 Gio, fichier référent de 100 Mio.
Version modifiée de `frag_find`, sans optimisation.

	Disque complet	1000	10000
Durée	1500 secondes	8 sec.	66 sec.
Volume lu	100 Gio	500 Kio	5 Mio
Proba faux négatif	0%	38.514645%	0.007181%

100 exécutions de la recherche avec 1000 échantillons :
Wins = 72 et **Fails = 28** $\Rightarrow$ 38,8% de faux négatifs...

Conclusions

1. E/S constituent le goulet d'étranglement. Augmentent significativement la durée de la recherche par échantillonage.
2. Adapter la taille de l'échantillonage à la probabilité acceptable de faux négatif

Plan

Références

- http://www.sans.org/reading_room/whitepapers/forensics/data-carving-concepts_32969
- http://www.cgsecurity.org/wiki/PhotoRec_FR
- http://www.dfrws.org/2010/
- http://www.dfrws.org/2010/proceedings/garfinkel1.pdf