Travaux pratiques

Format de fichier

Objectifs

Vous avez l'habitude d'utiliser nos équipements électroniques pour voir des images, photos, vidéos ou exécuter des programmes... Derrière toutes ces possibilités se cachent des fichiers. Nous nous intéresserons à comment un programme détecte le format d'un fichier pour le traiter. Connaître la structure d'un fichier est très important pour l'expert en sécurité car elle peut lui permettre de retrouver des informations utiles lors d'une analyse ou éviter des menaces. Au programme, nous verrons: *format, magic number, bombe de compression, loup-garou.*

Keywords and phrases Format de fichiers, métadonnée, caméléon, bombe de décompression
Digital Object Identifier TP1

1 Basique

1.1 Gestion de la machine virtuelle

Durant tout le cours, nous allons utiliser la distribution Kali Linux. Cette distribution est "la référence" en sécurité informatique pour réaliser des tests d'intrusion ou des audits de sécurité. Vous disposez d'une machine virtuelle avec Kali Linux pré-installé. Pour lancer la machine il suffit d'ouvrir un terminal et de lancer la commande suivante:

```
lance-vm-4MMSDSR.sh
```

Une fois que la machine sera lancée, vous vous authentifierez en tant qu'utilisateur `root` avec le mot de passe `toor`.

Vous allez avoir très certainement besoin de créer un répertoire partage entre le système hôte de la machine virtuelle.

En cas de problème au démarrage (ou même problème tout court) avec votre machine virtuelle, voici deux commandes qui feront le ménage et qui pourront peut être vous tirer d'affaire:

```
rm -rf "Virtualbox VMs"
rm -rf .config/Virtualbox
```

Si cela ne suffit pas, contactez le support support.info@ensimag.fr.

Si vous souhaitez répliquer cette configuration sur votre propre machine vous aurez besoin d'installer une machine virtuelle comme `VirtualBox` et de récupérer une image d'installation de Kali Linux. Si vous souhaitez aller plus loin avec Kali Linux, vous pouvez consulter leur guide https://kali.training/downloads/Kali-Linux-Revealed-1st-edition.pdf.

Pour chaque séance de travaux pratiques vous disposez d'une archive avec tous les fichiers nécessaires rangés (un répertoire par exercice).

1.2 Commandes utiles Unix

La Table 1 vous rappelle quelques commandes importantes d'Unix.

Une commande importante des système Unix/Linux est `dd`. Elle permet de copier des blocs de fichiers. Sa syntaxe est un peu particuliere mais facile a saisir:

Commande	Description
`pwd`	affiche le chemin du répertoire courant
`whoami`	affiche le nom de l'utilisateur
`history`	affiche l'historique des commandes
`ls`	liste le contenu d'un répertoire
`rm`	supprime un/des fichier(s) ou un/des repertoire(s)
`cp`	copie un/des fichier(s) ou un/des repertoire(s)
`mv`	renome/deplace un fichier ou un répertoire

Table 1 Commandes principales sous Unix.

```
dd if=dummy of=try bs=2K count=3 conv=sync,noerror
```

La commande précédente permet de copier `count`×`bs` octets du fichier `dummy` et de les écrire dans le fichier `test`.

2 Format de fichier

2.1 Principe

Un programme qui traite des données peut avoir besoin que celles-ci respectent un certain formatage. De même, un système d'exploitation a besoin de connaître le format d'un fichier pour choisir le programme adapté pour pourra le traiter sans systématiquement demander à l'utilisateur.
Pour identifier le format d'un fichier, on lui ajoute des métadonnées qui permettent de décrire le contenu du fichier. Il existe 2 types de métadonnées pour indiquer le format:

- les métadonnées externes (l'extension du nom de fichier ou comme pour les type `MIME`),
- les métadonnées internes (les *magic numbers*).

2.1.1 Métadonnées externes

On va modifier les données qui décrivent le fichier. La méthode la plus courante est celle qui consiste a modifier le nom du fichier pour lui ajouter une **extension**. Le nom du fichier est géré par le système de fichier associée au volume qui contient le fichier. Il s'agit donc bien d'une métadonnée externe. Cela a un avantage: pas besoin d'accéder réellement au fichier pour connaître son format.
On rajoute à la fin du nom du fichier une extension et on utilise le point `.` comme séparateur. Vous avez l'habitude de lancer des fichier `exe`, décompresser `zip`, écouter des fichier `mp3` et de compléter des fichier `docx` pour vos rapports ou compte-rendus. Les environnements Microsoft utilisent principalement les extensions pour choisir le programme approprié. Il existe 2 principaux types de fichier: les fichiers exécutables `exe` et les fichiers de données.
Les types MIME sont aussi des métadonnées externes permettant d'indiquer le format d'un contenu multimédia qui suit.

2.1.2 Métadonnées internes

Les fichiers que nous utilisons tous les jours contiennent une grande quantité de métadonnées internes comme nous le verrons plus tard. La première métadonnée que l'on trouve est le ***magic number***. Il s'agit d'un code permettant d'identifier le format du fichier au tout début. La table suivante vous donne quelques *magic numbers* pour des formats de fichier que nous allons étudier par la suite.

Magic number (en hexadécimal)	**Format**
42 5A 68	Fichier compressé par `BZIP2`
47 49 46 38 37 61 47 49 46 38 39 61	Fichier `ZIP`
FF D8 FF DB FF D8 FF E0 00 10	Image `JPEG`
4A 46 49 46 00 01	
50 4B 03 04 50 4B 05 06 50 4B 07 08	Fichier `ZIP`
7F 45 4C 46	Fichier `ELF`
25 21 50 53	Document Postscript
25 50 44 46 2D	Document `PDF`
1F 8B	Fichier compressé par `GZIP`

Exercice 1: commandes `xdg-open` et `file`

Dans les systèmes Linux, la commande `xdg-open` permet d'ouvrir automatiquement un fichier avec le programme approprié.

Question 1 Vous disposez dans le répertoire `EX1` de deux fichiers: `test` et `test.pdf`. Observer le comportement de `xdg-open`. Que pouvez vous conclure sur le fonctionnement de cette commande (penser à examiner les fichiers avec la commande `hexdump`) ?

On dispose aussi de la commande `file` qui permet d'identifier le format d'un fichier. Nous allons essayer de comprendre comment cette commande fonctionne.

```
dd if=/dev/urandom of=dummy bs=1 count=30
```

Question 2 Ajouter une extension de nom de votre choix au fichier `dummy`. Le résultat obtenu avec la commande `file` est-il consistant avec l'extension que vous avez ajoutée ?

Question 3 Ajouter un *magic number* au début du fichier `dummy`. La commande suivante va vous permettre d'ajouter une chaîne de caractères au début d'un fichier.

```
printf "HELLO" | cat - dummy > dummy.bis
```

Le résultat obtenu avec la commande `file` est-il consistant avec l'extension que vous avez ajoutée ? Quelle est votre conclusion sur la commande `file` ?

Exercice 2: Fuzzing

Le format d'archivage `tar` est très populaire dans le monde Unix et Linux. L'archivage permet de grouper dans une archive le contenu d'un ou plusieurs répertoires tout en respectant l'arborescence et les droits d'accès. La commande suivante illustre la création d'une archive contenant un fichier.

```
dd if=/dev/urandom of=dummy bs=50 count=1024
tar -cvf archive.tar dummy
```

Voici la commande qui va vous permettre de désarchiver le fichier.

```
tar -xvf archive.tar
```

Vous disposez d'une documentation qui va vous permettre d'avoir une description des métadonnées associées au format `tar`:

https://www.gnu.org/software/tar/manual/html_node/Standard.html

Question 4 Qu'est ce que permet d'identifier qu'un fichier est au format `tar` ? Combien de champs constitue l'entête d'un fichier `tar` ?

Question 5 Nous allons faire du *fuzzing* sur les différents champs qui composent le format `tar`. L'objectif est d'injecter des valeurs aléatoires dans les champs de l'entête pour voir si on obtient toujours des fichier `tar` valides. Pour cela vous disposez d'un début de script bash: `fuzzing.bash`. Pourquoi ce script ne pourra jamais produire de fichier `tar` valide tant que la seconde partie restera commentée ?

Question 6 Décommenter la seconde partie du script. Deux valeurs sont manquantes pour que le script fonctionne. Trouver ces valeurs et faites fonctionner les script. Combien de champs peuvent contenir des valeurs aléatoires et permettent d'avoir des fichier `tar` valides ?

Exercice 3: Caméléon

Un caméléon est un fichier qui a peut être détecté comme ayant plusieurs formats suivant le logiciel qui l'analyse. On ne peut pas faire des caméléons avec tous les formats de fichier. Le format d'archivage `tar` est très intéressant.
Question 7 En vous inspirant du script `fuzzing.bash`, écrivez un script `cameleon.bash` qui permet d'injecter un `magic number` dans le champs `name` de l'entête d'un fichier `tar`. Comment les fichiers que vous obtenez avec `cameleon.bash` sont ils détectées par la commande `file` ?

Question 8 Nous allons faire des tests avec un fichier un peu plus intéressant. Le fichier `eicar.com` permet de tester un anti-virus, vous pouvez le trouver dans `EX3`.

Créer une archive `eicar.tar` contenant `eicar.com`. Verser les fichiers `eicar.com` et `eicar.tar` sur agrégateur d'anti-virus de Google https://www.virustotal.com/fr/. Combien d'anti-virus détecte correctement `eicar.com` et `eicar.tar`.

Question 9 Créer un caméléon pour `eicar.tar` avec le *magic number* du format `gzip`. Combien d'anti-virus détecte correctement votre caméléon ?

3 Bombe de décompression et d'archivage

Une bombe de décompression ou d'archivage est un fichier de donnée malicieux qui a pour objectif de provoquer un déni de service. Plusieurs formes de déni de service sont possibles. On a tendance à ne pas distinguer les deux formes de bombes du à la dualité du format `zip` (qui fait à la fois de l'archivage et de la compression).

3.1 Décompression

Une bombe de décompression est un fichier compressé qui a un comportement anormal lors de la décompression. Une bombe exploite l'efficacité des algorithmes de compression: le fichier compressé est très petit mais quand il est décompressé on obtient un fichier énorme qui consomme toute la mémoire disponible. Le programme qui essaye d'ouvrir le fichier risque donc de planter.
La commande suivante permet de créer une bombe de décompression au format `gzip`.

```
dd if=/dev/zero bs=3MB count=100 | gzip -c > bomb.gz
```

```
ls -lhs bomb.gz
288K -rw-r--r-- 1 lauradou lauradou 285K sept. 19 23:29 bomb.gz
gunzip bomb.gz
ls -lhs bomb
287M -rw-r--r-- 1 lauradou lauradou 287M sept. 19 23:29 bomb
```

Il éxiste aussi des bombes qui exploitent des bugs dans les algorithmes de décompression pour avoir une décompression infinie. Une famille particulière de bombe de décompression est les *quines* (https://research.swtch.com/zip). Il s'agit de fichier compressé qui se décompressent en eux-même. L'idée est que si un programme essaye de les décompresser récursivement, il va le faire á l'infini.
La compression étant utilisée partout, il existe d'autres formats de fichiers qui peuvent être des bombes de décompression.

3.2 Archivage

Les bombes d'archivage ont pour but de pourrir la vie de l'utilisateur. Lorsque l'utilisateur va ouvrir l'archive, il va être submergé de fichiers. La bombe d'archivage peut aussi essayer d'écraser des fichiers existants via des chemins relatifs. Les dégâts peuvent être importants si par exemple la bombe réussi à écraser des fichiers sensibles comme `/etc/passwd`.

3.3 Format `zip`

Le format `zip` combine archivage et compression. C'est donc l'enfer en terme de sécurité. La bombe de décompression la plus célèbre est une bombe `zip` a fragmentation. Le fichier `42.zip` (http://www.unforgettable.dk/) se décompresse en 16 fichiers contenant 16 fichiers...jusqu'à obtenir des fichiers `zip` contenant un fichier 4.3GB. Il faut donc faire très attention quand on ouvre un fichier `zip`.

Exercice 4: Bombe de décompression

Question 10 Pourquoi ai je utilisé `/dev/zero` et non `/dev/urandom` pour créer ma bombe de décompression au format `gzip` ?

Question 11 Quel est le meilleur taux de compression que l'on peut avoir pour une bombe au format `gzip` ? (le taux de compression est le rapport de taille entre le fichier décompressé et le fichier compressé)

Question 12 Existe-t-il un moyen (une option de `gunzip`) qui permettrait de détecter une bombe sans la faire exploser ?

Question 13 Fabriquer une bombe de décompression pour le format `bzip2`. Quel est le meilleur taux de compression que l'on peut obtenir ?

Question 14 Dans le répertoire `EX4`, vous trouverez 2 images au format `png`. Sont elles faciles à ouvrir ?

Exercice 5: Surprise

Question 15 Dans le répertoire `EX5`, vous disposer d'un fichier `suite.tar`. Pour le désarchiver, exécuter la commande suivante:

```
tar Pxvf suite.tar
```

Que c'est-il passé ? Comment l'éviter ?

Exercice 6: Image, métadonnées et EXIF

Les images que nous visionnons tous les jours contiennent beaucoup d'informations en elle-même. Elles contiennent encore plus d'informations si on sait exploiter les données EXIF (*Exchangeable image file format*). La commande `exiftool` permet de visualiser certaines métadonnées de beaucoup formats de fichier. Cette commande est très facile à invoquer. La première commande permet d'afficher les coordonnées GPS associées à une image.

```
exiftool -gpslatitude -gpslongitude image.jpg
exiftool image.jpg
```

Cette commande permet aussi de modifier les champs des différentes métadonnées.

```
exiftool -artist=moi a.jpg
```

La commande suivante permet de purger toutes les métadonnées d'une image.

```
exiftool -all= foo.jpg
```

Question 16 Inspecter les 3 fichiers `jpeg` du répertoire `EX6`.