Audit d'une machine Linux compromise

Guide des bonnes pratiques dans l'acquisition de traces sur machine piratée

Principes de base :

- Ne pas faire de modifications sur le système en cours d'analyse
- Ne pas faire confiance aux outils installés sur le système en cours d'analyse
- Récupérer toutes les informations utiles :
  - Processus en cours d'exécution et les librairies utilisées
  - Fichiers ouverts
  - Sauvegarder l'image de la mémoire RAM + SWAP
  - Sauvegarder la liste des ports TCP/UDP ouverts
  - Récupérer les informations sur les partitions
  - Récupérer les informations sur les systèmes de fichiers
- Garder une trace des actions réalisées
- Sauvegarder ces informations sur un support externe, d'une manière fiable,
- S'assurer que les informations sauvegardées sont intègres
- S'assurer qu'aucune modification a été faite ou ne peut se faire sur les informations sauvegardées.

Nous allons détailler chaque principe :

Ne pas faire de modifications sur le système en cours d'analyse

Chaque processus modifie l'état du système : un processus se charge en mémoire, accède à des fichiers, charge des librairies, modifie les systèmes de fichiers, alloue de la mémoire...
Même une fois mort, un processus laisse des traces en mémoire RAM et/ou dans le SWAP si l'espace mémoire précédemment alloué au processus n'a pas été ré-alloué à un autre processus. En tenant compte de ce fait, on peut retrouver des traces des processus précédemment exécutés en mémoire, des commandes tapées, des connexions réseaux... en recherchant dans la RAM et le SWAP.
Il en est de même pour les systèmes de fichiers : exemple, une simple commande comme :
$ more /var/log/messages
modifie au moins :

- la table des processus, l'espace mémoire
- la date de dernier accès aux fichiers :
  - `/bin/more`
  - `/lib/libtermcap.so.2`
  - `/lib/i686/libc.so.6`
  - `/lib/ld-linux.so.2`
  - `/var/log/messages`
- le `~/.bash_history` (en fonction du shell)

car les librairies dynamiques chargées par la commande more sont :

```
$ ldd /bin/more
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40029000)
        libc.so.6 => /lib/i686/libc.so.6 (0x4002e000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Donc chaque action (commande tapée, création de fichier...) a un impact et peut effacer des données pouvant être utiles par la suite.

Ne pas faire confiance aux outils installés sur le système en cours d'analyse

Un système en cours d'analyse peut contenir des commandes, librairies, scripts modifiés... :

- Des commandes qui peuvent cacher des informations, peuvent lancer d'autres commandes ou envoyer des informations
- des librairies modifiées qui peuvent modifier le comportement des commandes sans que ces dernières soient modifiées
- des scripts qui peuvent par exemple lancer des services cachés au démarrage du système
- des modules du noyau peuvent masquer des informations issues de commandes sans que celles ci ou leur librairies associées soient modifiées

On ne peut pas faire confiance aux commandes du système sans être sûr que leur comportement n'est pas modifié. Il est nécessaire de recourir à des commandes ne faisant pas appel aux librairies systèmes (non dynamiques, mais linkés statiquement)

exemple : les commandes

« `ps -ef` » et « `top` » peuvent masquer des processus

« `ifconfig` » peut masquer le mode « PROMISC » des interfaces

« `ls` », « `find` » peuvent masquer la présence de certains fichiers

« `netstat` » peut masquer certaines connexions vers certaines adresses IP

Récupérer toutes les informations utiles

En sauvegardant le maximum d'informations sur l'état du système, on se donne toutes les chances de trouver des traces de compromission sur le système.

Plusieurs types d'informations caractérisent l'état d'un système :

- la table des processus lancés :
  - leur espace mémoire utilisé
  - le fichier exécutable qui a été chargé
  - les fichiers ouverts par ce processus (librairies, fichiers de données, devices)
  - les connexions ouvertes par ce processus
- l'état des interfaces réseau
- la liste des modules du noyau
- la table de routage
- la table arp (correspondance adresse ethernet et adresse IP)
- l'espace swap
- l'espace mémoire RAM
- l'espace disque (fichier -date de création, d'accès, de modification-, répertoires, inodes, blocks vides, fichiers effacés)

Un certain nombre de ces informations sont volatiles : processus, espace mémoire (RAM & SWAP), connexions ouvertes...

Donc avant de stopper un système à analyser, on peut sauvegarder :

- l'espace RAM
- l'espace SWAP
- la table des processus et le détail de chaque processus
- la table des connexions réseau
- l'état des interfaces réseau
- la table de routage
- la table arp
- la liste des fichies ouverts

Par la suite, on peut sauvegarder les systèmes de fichiers

Garder une trace des actions réalisées

Comment être sûr que le système n'a pas été modifié par notre analyse ?

Comment être sûr de ne rien avoir oublié ?

Comment avoir confiance en les informations que l'on recueuille alors même qu'on est en train de le modifier ?

Le fait de documenter les actions que l'on exécute fait que si un aspect n'a pas été traité lors d'une première analyse, on pourra tenir compte des conséquences supposées ou connues de ces actions lors d'une autre analyse.

Sauvegarder ces informations sur un support externe, d'une manière fiable

Où sauvegarder toutes ces informations ?

Sur le réseau ?

Sur un support connecté au système ?

Sur un support externe ?

Etre sûr que les informations ne vont pas subir de modifications lors des opérations de collecte, de transport ou de stockage.

S'assurer qu'aucune modification a été faite ou ne peut se faire sur les informations sauvegardées

Il existe des outils permettant de vérifier l'intégrité des informations sauvegardées. Les algorithmes de hashage MD5, SHA-1, SHA-256 peuvent être utilisés pour signer les informations extraites lors de l'extraction, puis par la suite, vérifier cette signature lors de l'analyse. Cela permet de s'assurer que les informations sauvegardées sont bien celles que l'on a extraites.

Répertoire des commandes

lister les partitions des disques
`fdisk -l /dev/sda` (pour les disques SCSI, `/dev/hda` pour les IDE)

/* mmls lecture d'une structure de disques */
```
mmls -t dos sda.dd : liste des partitions
dd if=sda.dd skip=x count=y of=sda1.dd
x=start y=lenght
```

vérifier l'heure

/ écoute sur le port 12345/TCP et enregistre les informations reçues dans le fichier fichier.txt) */*
```
$ nc -l -p 12345 > fichier.txt
```

/* table des addresses MAC */:
```
(remote)# nc -l -p port > arp_compromised
(compromised)# /mnt/cdrom/arp -an | /mnt/cdrom/nc (remote) port
(remote)# md5sum arp_compromised > arp_compromised.md5
```

/* Table des route */
```
(remote)# nc -l -p port > route_compromised
(compromised) # /mnt/cdrom/route -Cn | /mnt/cdrom/nc (remote) port
(remote)#md5sum route_compromised > route_compromised.md5
```

/* Table des modules du noyau */
```
(remote)# nc -l -p port > lkms_compromised
(compromised)#/mnt/cdrom/cat /proc/modules | /mnt/cdrom/nc (remote) port
(remote)# nc -l -p port > lkms_compromised.md5
(compromised)# /mnt/cdrom/md5sum /proc/modules | /mnt/cdrom/nc (remote) port
```

Phrack Magazine "Finding hidden kernel modules (the extrem way)".

The hunter.o module checks a chain of modules loaded into the kernel.

```
(compromised)#/mnt/cdrom/insmod -f /mnt/cdrom/hunter.o
(remote)#nc -l -p port > modules_hunter_compromised
(compromised)#/mnt/cdrom/cat    /proc/showmodules    &&    /mnt/cdrom/dmesg    |
/mnt/cdrom/nc (remote) port
(remote)#md5sum modules_hunter_compromised > modules_hunter_compromised.md5
(remote)#nc -l -p port > lsof_compromised
(compromised)#/mnt/cdrom/lsof -n -P -l | /mnt/cdrom/nc (remote) port
(remote)#md5sum lsof_compromised > lsof_compromised.md5
```

pcat (TCT)

```
(remote)#nc -l -p port > proc_id_compromised
(compromised)#/mnt/cdrom/pcat proc_id | /mnt/cdrom/nc (remote) port
(remote)#md5 proc_ip_compromised > proc_ip_compromised.md5
```

pcat PID | strings | more

exemple lsof

```
...
smbd       3137  root   20w   REG        8,1      253      46934
/var/log/httpd/access_log (deleted)
...

(remote)# nc -l -p port > ls_from_proc_3137
(compromised)# /mnt/cdrom/ls -la /proc/3137/fd/ | /mnt/cdrom/nc (remote) port
(remote)# more ls_from_proc_3137

l-wx------ 1 root root 64 Aug 10 21:03 12 -> /var/log/httpd/access_log (deleted)
...

(remote)# nc -l -p port > deleted_access_log
(compromised)# /mnt/cdrom/cat /proc/3137/fd/1 | /mnt/cdrom/nc (remote) port
```

/* File-modification times */

atime: ls –alRu /

ctime: ls –clRu /

mtime: ls –alR

/* Connexions réseau */

ifconfig -a

netstat –anp

/* Sauvegarde des fichiers */

utmp: current user access/accounting data

wtmp: historical user access/accounting data

lastlog: last access/login data

fichiers syslog :

/etc/syslog.conf

/var/adm/messages ou /var/log/messages

Application logs

Shell history files

Apache access_log

FTP xferlog

/* Sauvegarder les fichiers de configuration */

Authentication: /etc/passwd, /etc/shadow

Trust relationships: /etc/hosts.equiv,~/.rhosts

TCP wrapper rules: /etc/hosts.allow, /etc/hosts.deny

inetd/xinetd: /etc/inetd.conf, /etc/xinetd.conf, /etc/xinetd.d

Startup files: /etc/inittab, /etc/rc*

Scheduled events: /var/spool/cron/*

Find deleted files

lsof: list open files

Look for files named only by disk partition

Investigate /proc entries for processes

```
# ls -al /proc/1403
dr-xr-xr-x   3 root root 0 Apr 22 04:09 .
dr-xr-xr-x  78 root root 0 Apr 17 13:33 ..
dr-xr-xr-x   2 root root 0 Apr 26 19:25 attr
-r--------   1 root root 0 Apr 26 19:25 auxv
-r--r--r--   1 root root 0 Apr 26 19:24 cmdline
lrwxrwxrwx   1 root root 0 Apr 26 19:25 cwd -> /
-r--------   1 root root 0 Apr 26 19:25 environ
lrwxrwxrwx   1 root root 0 Apr 26 19:25 exe -> /sbin/syslogd
dr-x------   2 root root 0 Apr 26 19:25 fd
-r--------   1 root root 0 Apr 26 19:25 maps
-rw-------   1 root root 0 Apr 26 19:25 mem
-r--r--r--   1 root root 0 Apr 26 19:25 mounts
lrwxrwxrwx   1 root root 0 Apr 26 19:25 root -> /
-r--r--r--   1 root root 0 Apr 26 19:24 stat
-r--r--r--   1 root root 0 Apr 26 19:25 statm
-r--r--r--   1 root root 0 Apr 26 19:24 status
dr-xr-xr-x   3 root root 0 Apr 26 19:25 task
-r--r--r--   1 root root 0 Apr 26 19:25 wchan
```

Find deleted files (continued)

exe link points to binary image file for process

Can access binary via link even if file deleted

cmdline file contains command line args

```
cat cmdline
```

fd directory lists all open files by file descriptor

```
> ls -al fd
total 7
dr-x------  2 root root  0 Apr 26 19:25 .
dr-xr-xr-x  3 root root  0 Apr 22 04:09 ..
lrwx------  1 root root 64 Apr 26 19:25 0 -> socket:[2715]
l-wx------  1 root root 64 Apr 26 19:25 2 -> /var/log/messages
l-wx------  1 root root 64 Apr 26 19:25 3 -> /var/log/secure
l-wx------  1 root root 64 Apr 26 19:25 4 -> /var/log/maillog
l-wx------  1 root root 64 Apr 26 19:25 5 -> /var/log/cron
l-wx------  1 root root 64 Apr 26 19:25 6 -> /var/log/spooler
l-wx------  1 root root 64 Apr 26 19:25 7 -> /var/log/boot.log
```

/* enregistrer ses actions */

`history` liste des commandes tapées

script liste des frappes clavier et des sorties des commandes

```
script /mnt/floppy/log.txt
```

/* enregistrement des checksum etcomparaison */

md5sum /dev/hda == md5sum hda.img

/* duiplication d'un disque en local*/

dd `if=/dev/hda of=/mnt/disk/hda.img conv=noerror,notrunc bs=16384`

/* duiplication d'un disqueà travers le réseau (crypté) */

```
dd if=/dev/had conv=noerror,sync | des -e -c -k password | nc -w 3
    targetIP 2222
nc -l -p 2222 | des -d -c -k password | dd of=hda.img
```

/*script de duplication d'un disque en plusieurs fichiers */

```
#!/bin/bash
blocksz=20480k
let count=1
while(dd if=/dev/hda of=/mnt/disk/hda.$count.img bs=$blocksz skip=($
    ($count-1)) conv=noerror,notrunc)
do
      echo "Block $count output."
      count=$((count+1))
done
```

/ envoie le résultat de la commande 'ps -ef' sur la machine 192.168.0.x sur le port 12345 */*

```
$ ps -ef | nc 192.168.0.x 12345
```

/ dump de la RAM au format ELF -utilisable avec GDB */*

```
$ dd if=/proc/kcore | nc 192.168.0.x 12345
```

ou

/ dump de la RAM */ - la différence ????*

```
$ dd if=/dev/mem | nc 192.168.0.x 12345
```

/ donne la partition SWAP */*

```
$ fdisk -l /dev/hda | grep -i swap
/dev/sda2          1289      1353    522112+  82  Linux swap
```

/ dump du SWAP */*

```
$ dd if=/dev/sda2 | nc 192.168.0.x 12345
```

/ dump d'une partition quel que soit le filesystem : ext2|3, jfs, xfs, reiserfs...*/*

```
$ dd if=/dev/sda1 | nc 192.168.0.x 12345
```

/ dump d'un disque */*
```
$ dd if=/dev/sda | nc 192.168.0.x 12345
```

/ montage (en Read-Only) d'une partition sauvegardée par netcat : */*
```
$ mount -o loop=/dev/loop0,ro -t ext3 partition.dd /mnt/compromis
```

ou on peut l'analyser via le sleuthkit

image mémoire (prise par VMWARE) du système compromis :
```
     linux.vmss
```

- les ports ouverts :
```
$ netstat -anp
```
- sur les processus en cours d'exécution :
```
$ lsof -i
$ lsof -n
$ ltrace ou strace (trace des appels systèmes d'un binaire)
$ ldd (affichage des librairies dont dépends le binaire)
```
utilisation d'un détecteur de rootkit :
chkrootkit (chkproc) et rkhunter
`$ chkproc -v -u` : recherche des processus cachés dans 'ps'
comparaison de la liste des processus fournie par 'ps' avec la liste des PID dans /proc

/* Sleuhkit : fls : liste des fichiers du filesystem sans modifier l'atime */
```
            fls -f linux-ext3 -r -m / /dev/sda1 | nc -w3 192.168.0.x 12345
```

/* liste des fichiers de /dev */
```
grep « |\/dev\/ » fls.rm > fls.rm.dev
mactime-b fls.rm.dev >timeline.dev
grep « |\/\.» fls.rm > fls.rm.dot
mactime-b fls.rm.dot >timeline.dot
```

/* comparer la sortie des outils du système compromis et ceux qui sont propres */
```
netstat -anp
ifconfig -a
ps -eaxf
```

*/ lancer un nmap contre le systeme compromis en fin de récolte des donnés (ça laisse beaucoup de traces) */
```
nmap -sS -p 1- 192.168.1.79
```

/* comparaison de hash systemes avec ceux une hash database : */
Autopsy : hash database, il reste seulement les fichiers modifiés
types de fichiers : compress, data, exec, text
script perl

Analyses de la compromission :
- dans l'image RAM et dans le SWAP
```
$ strings linux.vmss | more
$ strings linux.vmss | grep "mot cle"
```
mots clé : http, get, kit, rootkit, .tar, .gz, .tgz rk ftp ...rm
grep -e “\d+\.\d+\.\d+\.\d+” = @IP
ou
```
$ strings -t -d vmss > string.vmss
```

```
$ grep wget string.vmss
```

/* rechercher */
dans .bash_history
les trous dans les logs
dans /proc

```
/proc/tcp : 5eme col = PID
/proc/PID/exe
/proc/modules
```

`/var/log/maillog` (certains rootkits envoient des messages)
`/var/log/secure`

lecture de /proc : utilitaires `procget procsave`
lecture RAM : memget

Détection de rootkit : `chkrootkit`

RFC 3227 : Guidelines for Evidence Collection and Archiving
http://www.ietf.org/rfc/rfc3227.txt