2008

UFR Ingénieurs 2000

Vivien Boistuaud

SECURITE DES SYSTEMES ET RESEAUX – MISE EN PLACE D'UN SYSTEME D'INFORMATION SECURISE

Ce document a été réalisé par V. Boistuaud dans le cadre des travaux pratiques de Sécurité des réseaux coordonnés par Sébastien Lelong et Yves Lavanant, au sein de l'UFR Ingénieurs 2000 de l'Université de Marne-la-vallée.

Table des matières

Introduction

Dans le cadre de mes études d'ingénieur en Informatique et Réseaux à l'université de Marne-la-vallée, j'ai bénéficié de cours en matière de sécurité des systèmes, des réseaux et des applications.

Deux séances de travaux pratiques ont été organisées : l'une portait sur la configuration sécurisée d'une machine Linux et la seconde portait sur les systèmes de détection d'intrusion et la sécurité applicative.

Ce rapport décrit la partie traitant de la configuration d'un système d'information sécurisé, et notamment de la mise en place d'une stratégie restrictive de protection d'un hôte d'un réseau.

A la lecture de ce document, il est essentiel de prendre en considération les points suivants : la sécurité d'un réseau se détermine par la sécurité de son composant le plus faible, qui est donc le plus facile à attaquer ; il vaut mieux être le plus restrictif possible pour sécuriser au mieux une machine : les besoins doivent être évalués à la suite de la sécurisation pour déterminer les souplesses de stratégie à adopter.

1. Mode opératoire et objectifs poursuivis

L'objectif de cette série de travaux pratiques en sécurité des systèmes et réseaux est de nous permettre d'apprendre à concevoir un système le plus sécurisé possible, pour qu'il puisse être utilisé avec le moins de risques possibles dans un environnement hostile (par exemple dans un environnement de production en entreprise).

1. Matériel mis en œuvre

Durant ces séances de travaux pratiques, nous avons utilisé le matériel suivant :

- Un ordinateur type PC, avec 512Mo de RAM et un disque dur4 Go,
- Un câble réseau pour la connexion au réseau Ethernet de l'école,
- Une connexion internet (toutefois pas indispensable).

Le but du TP étant d'apprendre à configurer une machine de façon sécurisée, comme si on configurait le serveur hébergeant les services Web d'une entreprise, il n'est pas nécessaire d'avoir plus de matériel pour achever nos missions.

La partie concernant l'installation de Debian GNU/Linux sur les machines ne sera pas abordée. Toutefois, si vous souhaitez reproduire ce TP à partir de rien, il vous faudra au moins un CD-ROM d'installation de Debian GNU/Linux. La version utilisée dans ce TP est Etch (la plus à jour des versions stables actuelles).

2. Logiciels mis en œuvre

La liste des logiciels mis en œuvre et la façon de les installer sera décrite au fur et à mesure du mode opératoire. La liste ci-après est fournie pour aide-mémoire uniquement :

- Commande Unix find,
- Apache 2.2 (Service http/https),
- Proftpd (service FTP),
- SSH (daemon,client et générateur de clefs),
- TCP Wrapper (xinetd)
- Iptables (firewall Linux 2.4/2.6)
- Snort (IDS Réseau)
- Nmap (Scanner de ports)
- Nessus (Audit de sécurité – agressif)
- OpenVPN (gestionnaire de connexions VPN)

2. Installation d'un système sécurisé

L'installation du système n'est pas abordée dans ce document. Dans le cadre du TP, le système Debian Etch a été installé automatiquement en utilisant BOOTP et en lançant l'installation par le réseau, depuis un serveur d'installation. Toutefois, ce type d'utilisation n'est pas conseillé : c'est ce que cette section explique.

1. Pré-requis à l'installation

Afin de configurer une machine de la façon la plus sécurisée possible, il est indispensable de la déconnecter du réseau avant de lancer l'installation.

En effet, la sécurité d'un réseau dépend de la sécurité du système le plus faible qui s'y trouve. Une machine qui est en cours d'installation ou qui vient d'être installé comporte de nombreuses failles de sécurités : il est impossible de contrôler les ports ouverts sur la machine et les démons en cours d'exécution, et le système ne comporte pas les dernières mises à jour de sécurité.

2. Décisions durant l'installation

Pendant l'installation du système, il est souvent proposé (c'est le cas pour Debian Etch) d'installer automatiquement certaines fonctionnalités. Il est déconseillé d'installer ces « bundles » fournis par le système.

En effet, la plupart du temps, ils installent des paquets souvent inutiles, qu'il est fastidieux de nettoyer par la suite. Pour des résultats optimaux, il faut installer le moins d'éléments possibles sur le système dans un premier temps, et ajouter ce qu'il manque dans la phase post-installation.

Le système installe déjà suffisamment de choses inutiles par défaut, dans de nombreux cas.

3. Actions consécutives à l'installation

Une fois que l'installation a été achevée, il faut commencer par nettoyer les paquets inutiles. A ce moment, il faut rester déconnecté du réseau.

Sur les systèmes qui nous étaient fournis, de nombreux logiciels étaient inutiles : une interface graphique X.11 était installée, ainsi que le gestionnaire de bureau Xfce, des outils comme `iceweasel`, `gedit`…

Pour lister les paquets installés sur le système, sous Debian, il suffit d'utiliser la commande `dpkg`, qui permet d'accéder au gestionnaire de paquets Debian :

```
# dpkg –l
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:
uppercase=bad)
||/ Name           Version        Description
+++-==============-==============-============================================
ii  adduser        3.63           Add and remove users and groups
ii  apt            0.5.28.6       Advanced front-end for dpkg
ii  apt-show-versi 0.08           lists available package versions with distri
ii  apt-utils      0.5.28.6       APT utility programs
ii  aptitude       0.2.15.9-2     terminal-based apt frontend
rc  at             3.1.8-11       Delayed job execution and batch processing
…
```

Les parquets listés ci-dessous sont, à priori, utiles au système, cependant, on a pu aussi trouver des parquets comme :

```
…
ii  gcc            3.3.5-3        The GNU C compiler
ii  gcc-2.95       2.95.4-22      The GNU C compiler
ii  gcc-3.0        3.0.4-7        The GNU C compiler.
ii  gcc-3.0-base   3.0.4-7        The GNU Compiler Collection (base package).
ii  gcc-3.3        3.3.5-13       The GNU C compiler
ii  gcc-3.3-base   3.3.5-13       The GNU Compiler Collection (base package)
…
```

Ceux-ci sont à priori à proscrire : un serveur de production n'est pas supposé être utilisé pour la compilation d'applications : au besoin, il est préférable de faire de la cross-compilation et de compiler sur une machine autre, prévue à cet effet, pour ensuite installer l'application sur le serveur sécurisé. En effet, si un hacker réussi à entrer sur le système, il est probable qu'il tente de compiler un *RootKit*[1] Aussi, tout élément lui facilitant le travail, comme la présence d'un compilateur ou des sources du noyau linux installé sur la machine, sont autant d'éléments qui lui feront gagner du temps, et qui le rendront donc plus difficilement détectable.

De façon générale, toute application ou tout service qui n'est pas utile ni pour administrer la machine, ni pour fournir les services pour lesquels le serveur est supposé agir, sont à proscrire.

Une fois le système nettoyé, il faut encore installer les mises à jour de sécurité. Au choix : l'entreprise possède un réseau sécurisé utilisateur, auquel cas vous pouvez relier la machine uniquement pour télécharger les mises à jour (de votre machine

[1] Kit permettant de jouir des droits administrateurs sur une machine, tout en étant masqué aux yeux de l'administrateur.

vers le serveur de mise à jour). Soit l'entreprise n'en possède pas, et les mises à jour doivent être installées par un autre moyen (CD-ROM par exemple).

Une fois seulement les applications mises à jour et les failles de sécurités connues comblées, vous pouvez **envisager** de connecter la machine au réseau de production.

Toutefois, il est déconseillé de le faire maintenant : nous n'avons pas configuré tous les services dont nous auront besoin pour contrôler la machine à distance, ni les services Web.

La section suivante décrit quelques règles simples mais utiles à appliquer pour protéger votre système.

3. Prévention des risques liés au système de fichiers

1. Rappels sur les fichiers sous Linux

Sous Linux, chaque fichier peut posséder plusieurs types de droit qui lui sont assignables, représentés sur 12 bits comme suit :

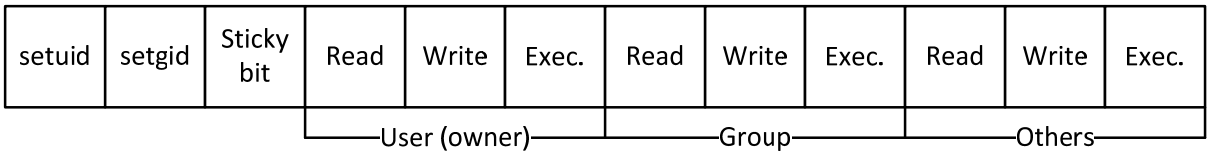

Figure 1 - Droits UNIX

Du fait que les droits peuvent être représentés comme 4 groupes de 3 bits, on utilise souvent la représentation octale (base 7) pour les représenter. Par exemple 4744 signifie que les bits suivants sont à 1 : setuid, user read, user write, user execute, group read, others read.

Les bits `setuid`, `setgid` et le `sticky bit` sont des valeurs qui doivent être utilisées avec parcimonie. En effet, ils permettent respectivement :

- D'exécuter une application (si utilisé avec droit Exec.) en bénéficiant des droits de l'utilisateur propriétaire,
- D'exécuter une application (si utilisé avec droit Exec.) en bénéficiant des droits du groupe propriétaire,
- De donner les droits pour écrire des fichiers dans un dossier (si le fichier sur lequel le droit est mis est un dossier) mais faire en sorte qu'un utilisateur ne puisse pas lire ni modifier les fichiers d'un autre.

Par conséquent, si un fichier dont le propriétaire est `root` (administrateur) ou dont le groupe propriétaire est le groupe administrateur, alors si le bit suid ou sgid est actif, un utilisateur ayant les droits d'exécution pourra exécuter un programme avec les droits root. Si le programme en question est modifiable, ou si c'est un shell, alors ceci peut donner des droits trop importants aux utilisateurs, et une personne mal intentionnée peut alors exécuter des codes malicieux sur la machine.

Par conséquent, il est nécessaire de vérifier que, sur le système, il n'existe pas de fichiers possédant des droits pouvant favoriser ce type de problème. De plus, il est important de vérifier régulièrement que ce type de fichier n'existe pas ou n'existe que de façon contrôlée.

Nous allons voir comment il est possible de détecter les fichiers présents sur le système en fonction des droits qui leurs sont accordés.

2. Vérifications courantes et utiles

Voici quelques exemples de commandes permettant de rechercher des informations utiles :

1. Recherche des fichiers porteurs du suid ou sgid

Pour chercher tous les fichiers du système qui ont le bit suid ou sgid activé, il suffit d'utiliser la commande :

```
# find / -perm 4000 –o –perm -2000 –ls
222806    2 drwxrwsr-x   2 root     staff        2048 oct 28  2006 /var/local
222815    2 drwxrwsr-x   2 root     mail         2048 sep 12 15:30 /var/mail
222470    2 drwxr-sr-x  16 man      root         2048 sep 12 15:32
/var/cache/man
222478    2 drwxr-sr-x   2 man      root         2048 sep 12 15:32
/var/cache/man/cat7
222480    2 drwxr-sr-x   2 man      root         2048 sep 12 15:32
/var/cache/man/cat9
222484    2 drwxr-sr-x   2 man      root         2048 sep 12 15:32
/var/cache/man/opt
222482    2 drwxr-sr-x   2 man      root         2048 sep 12 15:32
/var/cache/man/local
222477    2 drwxr-sr-x   2 man      root         2048 sep 12 15:32
/var/cache/man/cat6
…
```

La signification des arguments est la suivante :

- / => Recherche depuis la racine du système de fichier,
- -perm xxxx => permission recherché, si xxx commence par un tiret, alors ceci signifie que la chaîne recherchée est fixe, si ceci commence par un plus, alors ceci signifie « au moins une » des valeurs est vraie,
- -o => ou (OR)
- -ls => affiche les informations détaillées de chaque fichier (droits, propriétaire)

De tels fichiers peuvent potentiellement être dangereux, et il faut contrôler régulièrement leur existence, pour les raisons décrites dans la section précédente (section 2.1).

2. Recherche des fichiers en écriture pour « les autres » (others)

Pour chercher tous les fichiers du système dont le droit d'écriture aux autres est accordé, quel que soit l'utilisateur propriétaire, on peut utiliser la commande suivante :

```
# find / -perm -0002 –ls
222807    2 drwxrwxrwt   3 root     root         2048 jan 21 13:38 /var/lock
222837    2 drwxrwxrwt   2 root     root         2048 sep 18 09:36 /var/tmp
481445    0 srw-rw-rw-   1 root     root            0 jan 21 13:38
/var/run/acpid.socket
489445    0 srwxrwxrwx   1 avahi    avahi           0 jan 21 13:38
/var/run/avahi-daemon/socket
509889    0 srwxrwxrwx   1 root     root            0 jan 21 13:38
/var/run/dbus/system_bus_socket
 65587    0 -rw-rw-rw-   1 root     root            0 jan 21 14:01
/proc/1/task/1/attr/current
 65589    0 -rw-rw-rw-   1 root     root            0 jan 21 14:01
/proc/1/task/1/attr/exec
 65590    0 -rw-rw-rw-   1 root     root            0 jan 21 14:01
/proc/1/task/1/attr/fscreate
…
```

Ces fichiers peuvent être modifiés par tous les utilisateurs du système, qu'ils appartiennent ou non au groupe de l'utilisateur. Dans la mesure où ceci peut provoquer des problèmes d'intégrité des données de l'utilisateur, par la modification par un autre, alors ce comportement peut être dangereux.

3. Recherche des fichiers sans propriétaire

Sous Linux, chaque fichier est porteur du numéro d'utilisateur et du numéro de groupe à qui appartient le fichier. Si ces numéros ne sont pas associés à un nom d'utilisateur ou de groupe dans les fichiers /etc/passwd ou /etc/groups, alors on considère que les fichiers n'ont pas de propriétaires.

Le problème qui se pose est que ces fichiers sont quand même porteurs d'un numéro d'utilisateur, et ce numéro pourrait être réattribué lors d'une future création de compte ou de groupe. Par conséquent, un fichier sans utilisateur peut constituer un danget, puisqu'il peut se retrouvé assigné à un nouvel utilisateur.

Pour détecter ce type de fichier, et pouvoir sélectionner le traitement à leur appliquer (généralement leur affecter un utilisateur, ou supprimer les fichiers), alors il faut utiliser la commande :

```
pc0b002-14:~# find / \( -nouser -o -nogroup \) -ls
78139 2354 -rw-r--r--   1 150      150       2400358 sep 25 15:22
/opt/RTSJ_RI-1.1_alpha3-cdimage.bz2
 78138 2344 -rw-r--r--   1 150      150       2390486 sep 25 16:05
/opt/RTSJ_RI-1.1_alpha2-cdimage.bz2
 78148    8 -r--r--r--   1 500      500          6402 avr  4  2007
/opt/timesys/rtsj_ri_1.4/license
 78146 1820 -rw-r--r--   1 500      500       1855628 avr  4  2007
/opt/timesys/rtsj_ri_1.4/btclasses.zip
 78147    2 -r--r--r--   1 500      500          1586 avr  4  2007
/opt/timesys/rtsj_ri_1.4/copyright
 78145   74 -r--r--r--   1 500      500         72824 avr  4  2007
/opt/timesys/rtsj_ri_1.4/RTSJ\ 1_1\ alpha\ release\ notes.html
…
```

Il est important de faire cette recherche à chaque suppression d'un utilisateur, ou lors de chaque extraction d'une archive porteuse de droits UNIX (comme une archive tar) pour éviter ces allocations involontaires.

Dans l'exemple ci-avant, si par exemple on crée un utilisateur ayant pour UID 500, alors

3. Attributs additionnels

D'autres attributs (ou attributs étendus) peuvent aussi être affectés aux fichiers d'un système de fichiers Linux, mais ceci peut être variable en fonction du système de fichier sous-jacent (ext3fs dans notre cas).

Pour lister ces attributs étendus, il suffit d'utiliser la commande :

```
# lsattr test_file
---------------- test_file
```

Pour pouvoir les modifier, il est possible d'utiliser la commande `chattr`. Cette commande permet de régler les modes :

- a pour le mode « append only », où on ne peut qu'ajouter des informations dans un fichier, mais ne pas réécrire par-dessus ce qui est déjà contenu à l'intérieur, et sans pouvoir en lire le contenu. Utile pour des fichiers de logs.
- c pour le mode d'auto-compression des fichiers au niveau système, utile pour les gros fichiers auquel on souhaite compresser mais en continuant d'y accéder de façon transparente.

Par exemple, si on veut activer la compression automatique sur le fichier précédent :

```
# chattr +c test_file
--------c-------- test_file
```

De nombreux modes sont disponibles. Pour en avoir une liste complète, il faut consulter la page de manuel correspondante : `man chattr`.

4. Installation des services sur le serveur

Une fois que le système a été attentivement contrôlé et configuré, on peut installer les applications serveur qui permettront de fournir les services aux utilisateurs du serveur.

1. Installation du serveur Web Apache 2.2 (http)

1. Installation sur le serveur

Sous Debian GNU/Linux, il existe un paquet qui permet d'installer le serveur web Apache 2.2 facilement. Celui-ci se nomme `apache2` et s'installe comme suit :

```
# apt-get install apache2
```

Sous Apache, tout peut se configurer, y compris le modèle multi-thread à utiliser pour la gestion des requêtes concurrentes. Par défaut sous Debian, ce mode est `mpm-worker`. Pour des raisons d'efficacité, notamment lors de l'utilisation de traitements en PHP, nous devons changer le mode de gestion pour `mpm-prefork`. Pour installer ce mode de gestion des threads, sous Debian, il suffit d'installer le paquet `apache2-mpm-prefork` comme suit :

```
# apt-get install apache2-mpm-prefork
```

Une fois le serveur installé, il faut le sécuriser au maximum, comme indiqué dans la section suivante.

2. Démarrage du serveur web sous Debian

Un script de démarrage est installé avec le paquet Debian pour Apache 2.2, il se trouve sous le nom `/etc/init.d/apache2 [start|stop|reload|force-reload]`. Pour qu'il puisse s'exécutez, vous devez éditer ce script (avec `vi` par exemple) et placer la variable `NO_START` à `0` au lieu de `1`.

Cette manipulation est, à priori, nécessaire uniquement pour Debian, avec le paquetage Apache 2.2.

3. Notions de sécurisations d'un serveur Web

Plusieurs éléments peuvent affecter la sécurité de la machine ou du service web, et ceux-ci sont modulables.

Un des facteurs affectant la sécurité du serveur est l'utilisation injustifiée d'un module mal configuré ou, pire, contenant une faille de sécurité. Par défaut, la version 2.2 d'Apache fournit avec Debian active un certains nombre de modules, dont seulement quelques un sont utiles au bon fonctionnement du système. Une première mesure de prévention des risques consiste donc à désactiver les modules inutiles comme indiqué ci-après.

```
# a2dismod nommodule
```

Par exemple :

```
# a2dismod userdir
# a2dismod usertrack
…
```

Une fois les modules inutiles désactivés, on peut trouver un reproche à faire au serveur web : il donne trop d'informations sur les pages d'erreur concernant la configuration de sa machine hôte. Ceci peut constituer une faille, car il est possible de savoir quel est le système d'exploitation sous-jacent et quelle est la version exacte d'Apache, ce qui permet de cibler les failles à attaquer.

Aussi, pour sécuriser la machine, il faut réduire le niveau de verbosité des messages d'erreur. Pour cela, il faut modifier le fichier /etc/apache2/apache2.conf en changeant la ligne :

```
ServerSignature On
```

En

```
ServerSignature off
```

Et s'assurer que dans chacun des domaines virtuels (dans le dossier `/etc/apache2/sites-available`) la ligne n'existe pas, ou qu'elle est bien positionnée sur :

```
ServerSignature off
```

Enfin, un dernier élément, invisible à un utilisateur classique, mais qui permet d'obtenir des informations sur la version d'Apache et les modules chargés est la verbosité du serveur dans ses en-têtes http. En effet, si on effectue manuellement une requête web au serveur, on obtient une réponse comme suit :

```
# nc 127.0.0.1 80
HEAD / HTTP/1.1
Host: localhost

HTTP/1.1 302 Found
Date: Mon, 21 Jan 2008 13:27:54 GMT
Server: Apache/2.2.3 (Debian)
Location: http://localhost/apache2-default/
Content-Type: text/html; charset=iso-8859-1
```

Le serveur ne devrait pas communiquer toutes ces informations, qui pourraient être utiles pour un hacker. Pour les limiter, il faut modifier le fichier de configuration `/etc/apache2/apache2.conf` comme suit :

```
# Il suffit de passer ServerTokens au niveau Prod
ServerTokens Prod
```

On obtient bien le résultat escompté :

```
# nc 127.0.0.1 80
HEAD / HTTP/1.1
Host: localhost

HTTP/1.1 302 Found
Date: Mon, 21 Jan 2008 13:32:22 GMT
Server: Apache
Location: http://localhost/apache2-default/
Content-Type: text/html; charset=iso-8859-1
```

De plus, pour éviter que le design par défaut des pages d'erreur ne puisse être analysé pour trouver des éléments sur la version d'Apache ou du système d'exploitation, il peut être utile d'ajouter ses propres documents d'erreur. Pour ce faire, il faut soit déclarer globalement (dans apache2.conf), soit pour chaque hôte virtuel (dans `/etc/apache2/sites-available`), placer la ligne :

```
ErrorDocument 404 /err404.html
```

Le chemin du fichier d'erreur est relatif au répertoire racine DocumentRoot de la configuration générale ou de l'hôte virtuel.

Enfin, le fichier de configuration par défaut (apache2.conf) contient souvent des lignes inutiles mais potentiellement dangereuses, comme :

```
        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>
```

Ces lignes sont à supprimer : elles ne sont utiles qu'en cas d'utilisation de scripts CGI, ce qui peut provoquer une faille de sécurité potentielle en fonction des droits avec lesquels le démon Apache 2.2 s'exécute. Dans notre cas :

```
User www-data
Group www-data
```

Le démon apache et ses enfants s'exécuteront en tant qu'utilisateur www-data et non root, ce qui permet de garantir une certaine sécurité : tous les fichiers du système ne seront pas accessibles depuis le serveur web.

Enfin, par soucis de sécurité en cas de non présence d'une page d'index dans un répertoire, on peut désactiver le listing des fichiers qui est automatiquement réalisé par Apache en utilisant la directive :

```
Options –Indexes
```

Cette option peut être définie de façon globale, ou dans une section <Directory>.

Une fois le serveur Apache installé de façon sécurisé, il faut pouvoir fournir aux utilisateurs (éditeurs du site web) un moyen pour publier les fichiers sur le serveur. Ceci peut se faire à l'aide d'un démon FTP, ce qui est la façon la plus courante de procéder.

2. Installation du serveur ProFTPd

ProFTPd est un démon implantant le protocole FTP (File Transfer Protocol) en utilisant les comptes et droits UNIX pour ses connexions. En d'autres termes, il s'intègre tout particulièrement à la configuration du système. Il peut cependant être utilisé avec d'autres modes d'authentification (interconnexion avec un LDAP ou une base de données par exemple).

Pour installer ProFTPd, il suffit d'utiliser la commande suivante sous Debian :

```
# apt-get install proftpd
```

Afin d'assurer une plus grande sécurité, nous allons ajouter quelques limitations d'accès au service FTP en modifiant le fichier de configuration de ProFTPd (`/etc/proftpd/proftpd.conf`).

1. Eviter les excès de transparence

Par défaut, tout comme Apache, ProFTPd communique son numéro de version lorsqu'une connexion est initiée. Dans la mesure où ceci peut faciliter les opérations de hacking et les recherches de failles de sécurité, il est conseillé de le désactiver.

Pour cela, il suffit de modifier et/ou ajouter les directives suivantes au fichier de configuration :

```
ServerName                      "My Secure FTP"
ServerIdent                      Off
```

Le changement de nom du serveur permet d'éviter de laisser penser que le serveur n'a pas été configure, et peut également donner des informations sur le programme utilisé pour l'installation du logiciel. Il est donc recommandé de l'adapter.

2. Limiter les accès par utilisateur

Pour limiter les accès à une seule connexion par utilisateur, ce qui est en général suffisant, il suffit d'ajouter la directive suivante dans le fichier de configuration :

```
MaxClientPerUser                1
```

3. Créer un service d'upload public

Sur internet, les serveurs FTP permettant un accès anonyme contiennent souvent un dossier nommé « incoming ». Celui-ci est destiné à la mise à disposition par les utilisateurs de nouveaux fichiers, qui doivent ensuite être contrôlés et validés par les administrateurs du service FTP avant de les mettre à disposition de tous.

Aussi, il faut que les droits permettent la création d'un fichier, mais pas la lecture ni la modification après upload, pour éviter les modifications d'intégrité. Pour cela, nous avons réalisé la configuration suivante :

```
<Anonymous /home/ftp>
        User                            ftp
        Group                           nogroup
   # We want clients to be able to login with "anonymous" as well as "ftp"
        UserAlias                       anonymous ftp
   # Cosmetic changes, all files belongs to ftp user
        DirFakeUser                     on ftp
        DirFakeGroup                    on ftp

    RequireValidShell           off

    # Limit the maximum number of anonymous logins
    MaxClients                  10

#   # We want 'welcome.msg' displayed at login, and '.message' displayed
#   # in each newly chdired directory.
    DisplayLogin                        welcome.msg
    DisplayFirstChdir           .message

#   # Limit WRITE everywhere in the anonymous chroot
    <Directory *>
      <Limit WRITE>
        DenyAll
      </Limit>
    </Directory>

#
#   # Uncomment this if you're brave.
   <Directory incoming>
      # Umask 022 is a good standard umask to prevent new files and dirs
      # (second parm) from being group and world writable.
      Umask                                 022  022
      <Limit READ WRITE>
         DenyAll
      </Limit>
      <Limit STOR>
         AllowAll
      </Limit>
   </Directory>
</Anonymous>
```

Remarque : il faut aussi penser à créer le dossier /home/ftp/incoming pour que les fichiers puissent être placés correctement sur le disque.

4. Empêcher l'accès à certains utilisateurs

Les utilisateurs dont le login est présent dans le fichier `/etc/ftpusers` sont interdits d'accès au service FTP. Pour empêcher le login d'un utilisateur donné, il suffit donc d'y ajouter le login correspondant. Tous les comptes systèmes non associés à un utilisateur physique devraient y être listés.

3. Installation du serveur SSHd et expérimentations SSH

Pour pouvoir administrer la machine à distance une fois qu'elle sera connectée au réseau, nous auront besoin d'ouvrir un shell distant. Il existe plusieurs méthodes pour cela : la plus ancienne, telnet, fait transister les informations en clair sur le réseau, ce qui peut être compromis par une attaque de type sniffing.

La seconde solution, développée pour OpenBSD dans les années 90 se baptise OpenSSH : Open Secure SHell. Les communications sont alors chiffrées et éventuellement compressées.

Sous Debian, OpenSSH est généralement pré-installé. S'il ne l'est pas, il suffit d'utiliser la commande suivante pour l'installer :

```
# apt-get install openssh-server
```

Pour les outils clients (à ne pas installer sur un serveur, mais seulement sur une machine Linux cliente) on peut installer le paquetage `openssh-client`.

Le fichier de configuration du serveur SSH se nomme par défaut `/etc/ssh/sshd_config`. Pour que la configuration prenne effet, il faut redémarrer le démon SSH : /etc/init.d/sshd restart. Ceci ne coupe pas les connexions actives.

1. Limiter les utilisateurs autorisés

La première mesure de sécurisation de SSH à prendre est de désactitver l'accès SSH au compte amdinistrateur (`root`). En effet, en cas d'attaque par brute force et de réussite de l'attaque, il est préférable que le compte hacké soit celui d'un utilisateur standard et non celui de l'administrateur. Ainsi, le hacker sera plus facilement détectable si, une fois logué, il tente un brute force sur la machine pour obtenir les droits d'administration.

Pour désactiver la connexion SSH en tant que root, il suffit de modifier la directive suivante dans la configuration :

```
PermitRootLogin no
```

De plus, tous les utilisateurs n'ont pas besoin d'accéder au shell distant. Pour restreindre l'accès à une liste des utilisateurs accrédités, il suffit d'ajouter la directive suivante :

```
Allowusers tata titi
```

Ainsi, tout autre utilisateur que tata et titi ne pourront pas se connecter. Notamment, les utilisateurs du FTP qui ne sont ni tata ni titi (par exemple, toto).

2. Authentification par clef publique

Un mot de passe peut être facilement attaqué : à l'aide d'un dictionnaire ou par brute force, aucun mot de passe ne peut s'avérer inviolable.

SSH propose donc une méthode beaucoup plus sûre, utilisant un cryptage asymétrique pour l'authentification de l'utilisateur. Le cryptage asymétrique peut être en RSA ou en DSA (le plus sûr) et la clef fait couramment 2048 bits.

Pour utiliser ce type d'authentification, il faut que chacun des utilisateurs génère d'abord une paire de clefs publiques/privées à l'aide de la commande :

```
ssh-keygen -tdsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/toto/.ssh/id_dsa)
```

Une fois les clefs créées, il faut transférer la clef publique vers le serveur sur lequel on souhaite utiliser une authentification par clef-publique. Par défaut, il faut que la clef soit ajoutée dans le fichier `.ssh/authorized_keys` de l'utilisateur auquel elle appartient. Ce transfert peut se faire, par exemple, par `scp` (SSH Copy).

Il faut également que ce type d'authentification soit autorisé dans la configuration du serveur ssh :

```
PubkeyAuthentication yes
```

Une fois les clefs des utilisateurs autorisés installées, il est possible de supprimer complètement l'authentification par mot de passe pour n'utiliser que l'authentification par clef publique :

```
PasswordAuthentication no
```

Le serveur n'est administrable à distance que de façon sécurisée.

3. Tunneling SSH

Là où SSH va plus loin que telnet, réside également en sa capacité à pouvoir créer des tunnels permettant le chiffrement de transmissions et la connexion à des services LAN hébergés sur un même réseau que la machine à laquelle on se connecte en SSH.

Par exemple, il arrive fréquemment en entreprise que seul un accès SSH soit autorisé au réseau en production, mais qu'il soit utilisé pour accéder à un serveur Web de monitoring qui n'est pas publiquement accessible de l'extérieur.

Pour permettre par exemple un accès sur le réseau local 10.2.28.0/24 à un serveur web hébergé à distance (adresse IP 10.2.29.0/24) via un tunnel SSH, si on considère qu'on possède deux machines 10.2.28.4/24 sur le réseau 1 et 10.2.29.5/24 sur le réseau 2, accessible publiquement par l'adresse 10.10.10.28/32, on utilisera la commande :

```
ssh -N -g -f -L 10.2.28.4:8080:10.2.29.5:80 toto@10.10.10.28
```

Ainsi, toute connexion à 10.2.28.4 sur le port 8080 redirigera la requête vers la machine 10.2.29.5 sur le port 80, en passant par la passerelle 10.10.10.28.

Les commutateurs utilisés ont la signification suivante :

- -N => Pas de shell interactif à lancer côté passerelle
- -g => Autoriser les connexions sur le port bindé par le client par les autres machines du réseau
- -f => Place l'application cliente ssh en arrière plan
- -L => Description de la redirection de port

4. Protections contre le spoofing

Pour éviter les attaques de type « Man in the Middle », SSH inclus un système de vérification de la clef des hôtes connus à chaque connexion : lors de la première connexion, l'utilisateur doit confirmer le footprint (empreinte) de la clef fournie par le serveur. S'il la valide, lors des connexions suivantes, l'emprunte de la clef fournie par le serveur sera comparée à celle stockée dans le cache des hôtes connus.

Si l'emprunte ne correspond pas, le message suivant sera affiché :

```
# ssh toto@127.0.0.1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
c6:2c:d1:55:38:af:26:fe:02:69:bc:1b:df:cc:97:bd.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending key in /root/.ssh/known_hosts:3
RSA host key for 127.0.0.1 has changed and you have requested strict checking.
Host key verification failed.
```

SSH permet d'administrer à distance et de façon sécurisée un système UNIX : cette solution a été adaptée par tous les systèmes UNIX (Solaris, HP-UX, Linux…) et en fait une technologie éprouvée et reconnue pour sa sécurité.

4. Utiliser Apache avec un module SSL (https)

Cette section n'est pas détaillée dans ce document. Nous vous conseillons de vous référer au document *« Configuration d'Apache », Vivien Boistuaud, Fabien Bidet, IR1*.

Une fois que les services utiles aux utilisateurs du serveur sont installés, il faut sécuriser la machine ainsi que les accès à ces services. La section suivante aborde cet aspect et, plus généralement, la configuration d'un firewall sous Linux.

5. Sécurisation des services

Dans les sections précédentes, nous nous sommes concentrés sur les opérations de sécurisation de base des services Web, FTP et SSH, ainsi que les fichiers à surveiller particulièrement sur un système fraîchement installé.

Cette section présente une façon plus avancée de protéger une machine UNIX face aux attaques extérieures.

1. Tuning du noyau Linux

Le noyau Linux propose une interface d'accès par fichiers virtuels à des options de configuration courantes du noyau, notamment celles de la couche IP (v4 dans notre cas). Cette section détaille quelques options utiles à activer ou désactiver.

1. Bloquer les ICMP ECHO REQUEST

Les messages ICMP ECHO (ceux envoyés par la commande ping) sont souvent utilisés pour détecter l'existence d'une machine sur un réseau. Un hacker cherchant une machine à attaquer commence, en général, par détecter sa présence à l'aide de ce type de paquets.

Linux permet, dans la configuration de son noyau, de désactiver ce type de sollicitations. Pour déasctiver uniquement les ping broadcast, on peut utiliser la commande :

```
# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Tandis que pour interdire tous les types de ping, on utilisera la commande :

```
# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

En réponse à ces actions, si on essaye de pinger la machine, aucune réponse, ni même erreur, ne sera transmise en réponse à un message ICMP ECHO REQUEST :

```
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 2999ms
```

Par conséquent, la plupart des scripts automatisés de recherche de machines sur un sous réseau ne détecteront pas l'existence du serveur, ce qui le protège de certaines attaques les plus courantes, type insignifiants (script kiddies).

2. Désactiver l'IP-forwarding

Le noyau Linux implante en standard la redirection de paquets (IP forwarding). Cette fonctionnalité est très intéressante pour transformer une machine classique en routeur, mais peut s'avérer dangereux dans d'autres cas.

En effet, la redirection s'effectue au niveau des couches 3 (réseau) et 4 (transport), ce qui a pour conséquence de ne pas provoquer le traitement des couches supérieures. Or, dans le cas de l'utilisation d'un firewall mandataire, qui n'analyse que les paquets au niveau de la couche 7, l'activation de la redirection de paquets aura pour effet de bypasser (contourner) le firewall, provoquant ainsi son inefficacité.

Pour désactiver l'IP-forwarding, il suffit de modifier le fichier virtuel correspondant comme suit :

```
# echo 0 > /proc/sys/net/ipv4/ip_forward
```

3. Désactiver les timestamps TCP

Les paquets TCP contiennent des marques de temps, qui servent de numéros de séquence. Ceux-ci peuvent être intéressants pour les hackers qui pourraient utiliser ces informations pour forger des paquets, en fonction de l'algorithme de génération sous-jacent.

De plus, les TCP Timestamps permettent de connaître l'uptime du serveur ou de la station Linux., ce qui est, comme indiqué précédemment, un moyen fiable de forger de faux-paquets.

Pour désactiver les timestamps TCP, il suffit d'utiliser la commande :

```
# echo 0 > tcp_timestamps
```

Ces quelques règles simples permettent néanmoins d'améliorer la sécurité contre les hackers, vis-à-vis du système même.

2. Augmenter la sécurité des mots de passe

Par défaut, sous Linux, les mots de passe sont libres : il suffit qu'ils fassent plus de 4 caractères environ et n'ont pas de règles d'assignation. Les utilisateurs étant généralement assez simplistes, ils choisissent des mots de passe facilement attaquables, ou qui sont trop rarement modifiés.

PAM est un module d'authentification pour Linux qui permet, entre autre, de définir de façon moins permissive les règles liées aux mots de passe des utilisateurs, sauf pour l'utilisateur root qui n'est soumis à aucune restriction. On considère que l'administrateur doit être assez intelligent pour choisir un mot de passe assez difficile à trouver.

Pour installer PAM avec la cracklib sous Debian, il faut utiliser la commande :

```
# apt-get install libpam-cracklib
```

Les fichiers permettant de personnaliser les paramètres de PAM se situent dans le dossier /etc/pam.d par défaut. Le fichier qui nous intéresse, pour modifier les règles de mot de passe, est le fichier common-password, auquel nous ajoutons la ligne :

```
password required  pam_cracklib.so retry=3 minlen=8 dcredit=-1 ocredit=-1
ucredit=-1 lcredit=-1
```

Cette ligne précise des règles spécifiques au module CrackLib de PAM comme suit:

- retry=3 # précise que les tentatives de saisie ne peuvent être faites que trois fois d'affilé tout au plus,
- minlen=8 # précise que la longueur minimale du mot de passe est 8 caractères
- dcredit=-1 # Il faut au moins un chiffre (décimal)
- ocredit=-1 # Il faut au moins un caractère non alphanumérique
- ucredit=-1 # Il faut au moins une majuscule
- lcredit=-1 # Il faut au moins une minuscule

Si les valeurs de Xcredit=n sont négatives, ceci signifie (comme c'est le cas ci-dessus) : « au moins abs(n) caractères de ce type ». Si le crédit est positif, il représente une pondération : une valeur de 2 signifie que le type de caractère compte comme deux caractères pour le calcul de la longueur minimale.

3. Utiliser TCP Wrapper pour les services peu sollicités

Lorsqu'un démon TCP/IP est peu sollicité, il est inutile d'avoir plusieurs processus le représentant qui occupent la table des processus. Le concept de TCP Wrapper apporte une solution à ce problème : plutôt que d'avoir un processus par service, on a un processus unique qui écoute plusieurs ports TCP/IP et qui, lorsqu'il reçoit une demande de connexion sur un port donné, lance le programme qui lui est associé.

A l'origine, sous Linux, le démon inetd assure ce travail. Cependant, il existe une version dite étendue baptisée xinetd et qui est celle que nous avons utilisé durant ce TP.

1. Installation de xinetd

Sous Debian, installer xinetd se fait simplement avec la commande suivante :

```
# apt-get install xinetd
```

2. Associer un nouveau service à xinetd

Les fichiers de configuration de xinetd se trouvent par défaut, sur Debian, dans le dossier /etc/xinetd.d/. La configuration est découpée en plusieurs fichiers, qui sont lus lors de l'initialisation du démon xinetd.

Pour associer notre service ProFTPd à xinetd, il faut dans un premier temps désactiver le lancement automatique de ProFTPd. Pour cela, sous Debian, il suffit de faire :

```
# dpkg-reconfigure proftpd
```

Et de choisir le mode de configuration « inetd ». Notez que cela ne créera pas automatiquement la configuration pour xinetd. Nous allons donc manuellement effectuer cette modification.

Pour cela, nous créons un nouveau fichier dans /etc/xinetd.d baptisé ftp et qui contient le code suivant :

```
service ftp
{
    flags         = REUSE
    socket_type   = stream
    wait          = no
    user          = root
    server        = /usr/sbin/proftpd
    disable             = no
}
```

De plus, pour que cette configuration fonctionne, il faut que le service ftp soit bien déclaré et ne soit pas commenté dans le fichier /etc/services.

Après redémarrage de xinetd, via /etc/init.d/xinetd restart, le service FTP est disponible.

3. Ajouter des restrictions à un service

Pour illustrer l'efficacité du démon xinetd en termes de pré-filtrage des communications TCP, nous avons ajouté deux nouvelles règles.

La première ne permet la connexion au service FTP que de 8h00 à 12h00 et de 13h00 à 18h00, heures d'ouvertures d'une entreprise par exemple. Pour cela, dans le bloc « service ftp » précédent, il suffit d'ajouter une ligne contenant :

```
access_times        = 8:00-12:00 13:00-18:00
```

La seconde que nous avons appliquée est une restriction par adresse IP de la machine autorisée à se connecter au service. Nous n'avons autorisé les connexions que depuis l'adresse IP 172.17.4.4 à l'aide de la ligne suivante ajoutée au bloc de service ftp :

```
only_from      = 172.17.4.4/32
```

Notez que, bien qu'on ne parle ici que de wrapping TCP, il est possible d'utiliser xinetd pour les services UDP, et d'utiliser des options plus avancées (logging par exemple des informations de connexion).

4. Firewalling

Les systèmes Linux 2.4 et 2.6 sont équipés d'un puissant firewall baptisé IPTables, issu du projet NetFilter, et qui peut être utilisé comme firewall stateless ou stateful.

Afin de démontrer la puissance de cet outil, j'ai réalisé en TP un script de configuration automatique d'iptables qui permette d'illustrer à la fois des règles stateless et stateful. Ce script, commenté, est le suivant :

```
#!/bin/bash

iptables -P INPUT ACCEPT            # Règle INPUT par défaut – tout accepter
iptables -P OUTPUT ACCEPT           # Règle OUTPUT par défaut – tout accepter
iptables -P FORWARD ACCEPT          # Règle FORWARD par défaut – tout accepter
iptables –F                         # Supprimer toutes les règles existantes
iptables –X                         # Supprimer les chaines additionnelles

# Tout interdire par défaut
iptables -P INPUT DROP
iptables -P OUTPUT DROP

# Accepter trafic local (loopback)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Accepter connexions HTTP entrantes
iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -j ACCEPT

# Accepter connexions HTTPS entrantes
iptables -A INPUT -i eth0 -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 443 -j ACCEPT

# Accepter connexions FTP depuis une seule adresse
iptables -A INPUT -i eth0 -p tcp -s 172.10.2.4 --dport 21 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp -d 172.10.2.4 --sport 21 -j ACCEPT

iptables -A INPUT  -i eth0 -p tcp -s 172.10.2.4 --dport 20 -j ACCEPT
iptables -A OUTPUT -i eth0 -p tcp -s 172.10.2.4 --sport 20 -j ACCEPT

# Règles stateful pour autoriser le traffic machine > internet

iptables -A INPUT -i eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j
ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 443 -m state --state ESTABLISHED -j
ACCEPT

iptables -A OUTPUT -o eth0 -p tcp --dport 80 -m state --state NEW -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 443 -m state --state NEW -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp --dport 53 -m state --state NEW -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 53 -m state --state ESTABLISHED -j
ACCEPT
```

Il suffit alors d'exécuter le script et de le tester pour vérifier qu'il fonctionne : on a accès à internet, et la machine autorisée à accéder au FTP à accès au FTP et au http.

IPTables fournit également deux outils pour sauvegarde et charger une configuration du firewall précédemment définie.

Pour sauvegarder, on utilise : `iptables-save > firewall.sav`

Pour recharger, on utilise : `iptables-restore < firewall.sav`

Enfin, IPTables permet également de faire du masquerading, que nous n'avons pas utilisé dans notre script. Ceci correspond en fait à la technologie NAT (Network Address Translation), qui permet de faire de la translation d'adresse entre plusieurs réseaux, dans le cas où la machine est utilisée comme routeur, ce qui n'est pas notre cas.

Les sécurités décrites précédemment permettent de restreindre l'accès aux services et/ou de les moduler selon certaines conditions. Toutefois, ceci ne nous permet pas de détecter les intrusions de hackers sur le système.

Pour se prévenir contre les attaques, ou au moins pour les détecter, il est important d'utiliser des systèmes de détection d'intrusion (IDS). C'est de ce sujet que traite la section suivante du présent document.

6. Les systèmes de détection d'intrusion

Un système de détection d'intrusion, ou IDS en anglais pour *Intrusion Detection System*, est un outil de protection permettant de détecter les intrusions potentielles dans un système.

Le but d'un IDS n'est que de détecter l'intrusion, il ne permet pas de la prévenir ni de s'en protéger. Cependant, la détection peut permettre d'envisager une riposte, le lancement d'un autre outil de traitement (comme un anti-virus ou un détecteur de rootkit, …).

Deux types d'IDS existent :

- Les IDS systèmes, dont le but est de détecter les intrusions sur un système hôte en vérifiant les modifications non autorisées du système et de ses fichiers.
- Les IDS réseaux, dont le but est de détecter les tentatives d'intrusion sur un réseau, comme le trafic de paquets attaquant une faille particulière d'un logiciel réseau.

Durant ce TP, nous avons testé ces deux différents types d'IDS de sorte à en valider le fonctionnement.

5. Fonctionnement d'un IDS Système

Un IDS système est un logiciel de détection des intrusions sur un système. En général, il est à installer sur des machines demandant un niveau de sécurité élevé.

Les plus connus sous Linux sont tripwire et AIDE. En aucun cas, un IDS ne permet de savoir quel RootKit a été installé, mais il vérifie que les fichiers systèmes couramment attaqués ne sont pas modifiés.

Installer tripwire sous Debian se résume à exécuter :

```
# apt-get install tripwire
```

Puis à personnaliser la liste des fichiers à surveiller. Les clefs nécessaires (clef locale et clef de site) seront générées sur demande à l'utilisateur. Si l'utilisateur possède déjà une clef de site générée, il est possible de la préciser manuellement par la suite.

6. Fonctionnement d'un IDS Réseau

Un IDS Réseau est un logiciel permettant de sniffer le réseau en vue de la détection d'attaques... Les plus connus sous Linux sont Prelude et Snort. Pour installer snort sur Debian, ainsi que les règles de filtrage par défaut, il suffit d'exécuter la commande suivante :

```
# apt-get install snort
```

L'adresse du réseau que nous souhaitions surveillé est : 172.17.0.0/16 car l'IP de la machine depuis laquelle les expérimentations ont été menées est 172.17.2.13.

Une fois snort installé, nous pouvons l'exécuté dans plusieurs modes : snort -v[de] pour afficher les informations plus ou moins détaillées sur les paquets.

Pour surveiller par exemple la réception du message « Hacking » depuis n'importe quelle machine à destination du réseau auquel notre machine appartient, sur n'importe quel port, il suffit d'ajouter la règle suivante au fichier `/etc/snort/rules/local.rules` (prévu pour contenir les règles personnalisées du système) :

```
alert tcp any any -> 127.17.0.0/16 any (content:"Hacking"; msg: "Je
t'y prend a parler de hacking")
```

Sous Debian, par défaut, snort fonctionne en mode daemon et logue ses messages dans le syslog. Pour vérifier que le message est reçu, il suffit alors de lancer une recherche permanente sur le fichier /var/log/syslog comme suit :

```
# tail –f /var/log/syslog
```

De plus, de nombreuses règles par défaut sont fournies dans /etc/snort/rules/, détectant la plupart des failles connues des principaux scripts pour le web, des failles FTP, SMTP…

7. Auditer la sécurité d'un système

De nombreux outils existent pour auditer la sécurité d'un système et, notamment, la perception qu'on a de lui depuis l'extérieur. Dans le cadre de ce TP, nous avons utilisé deux outils différents : nmap, qui est un scanner de port et permet de déterminer de façon probabiliste le système hôte ; et Nessus, qui est un testeur de vulnérabilité.

4. Utilisation d'un scanner de ports

Un scanner de port est un outil qui permet de déterminer quels ports sont ouverts sur une machine distante, et tente de détecter le type de communication qu'ils attendent. Ce type d'outil est pratique à la fois pour mener des audits de sécurité et pour commencer la recherche d'un système vulnérable pour un hacker, un cracker, ou un insignifiant.

Si nmap n'est pas déjà installé, sous Debian, il suffit d'utiliser la commande suivante pour l'installer :

```
# apt-get install nmap
```

Une fois nmap installé, il suffit de le lancer en précisant soit : le nom d'une machine, son adresse IP, ou l'adresse IP d'un réseau à scanner.

Par défaut, nmap s'arrête si la machine ne répond pas aux requêtes ICMP ECHO REQUEST, ce qui est le mieux si on souhaite scanner un réseau. Toutefois, si on est certain de l'existence d'une machine, alors utiliser nmap avec l'option –P0 (No ping) semble plus adapté.

Par exemple, vers le serveur www.ccscript.net, on obtient la liste de services suivante :

```
# nmap www.ccscript.net

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2008-02-19 03:22 CET
Interesting ports on rbx1-2.net.ccscript.net (91.121.33.20):
Not shown: 1674 closed ports
PORT     STATE    SERVICE
25/tcp   filtered smtp
80/tcp   open     http
135/tcp  filtered msrpc
445/tcp  filtered microsoft-ds
1720/tcp filtered H.323/Q.931
6667/tcp filtered irc

Nmap finished: 1 IP address (1 host up) scanned in 4.585 seconds
```

Les ports notes comme « filtered » sont les ports pour lesquels une réponse de rejet a été reçue. En l'occurrence, ceci provient du fait que le fournisseur de services (OVH) filtre un certain nombre de ports avant l'accès aux serveurs dédiés qu'il héberge.

5. Utilisation d'un scanner de vulnérabilités

Nessus est un puissant outil permettant de scanner les vulnérabilités d'un système. Son utilisation est à prescrire avec parcimonie, dans la mesure où une machine mal configurée ou sujette à des failles de sécurité pourrait être mise hors service par son utilisation.

Il fonctionne sur un modèle client-serveur, le client étant en fait une télécommande permettant de planifier les audits (ou attaques) pour les transmettre au serveur.

En conséquence, Nessus est également assimilable à un cheval de troie, bien que son utilisation pour tester la sécurité de ses propres systèmes soit très efficace.

Pour installer Nessus sur Debian, il suffit d'utiliser la commande :

```
# apt-get install nessus nessud
```

Ainsi, on installe le client et le serveur sur la même machine. Pour des audits demandant un débit important, on pourra installer le client sur la machine locale de l'auteur de l'audit, et le serveur sur une connexion plus adaptée.

Malheureusement, je n'ai pas fait de copies d'écran de Nessus durant ce TP. Cependant, dans la mesure où nos systèmes étaient à jour et avaient été préalablement sécurisé, le rapport de Nessus ne contenait pas d'erreurs importantes ni d'informations sur des failles potentielles.

7. Réseaux privés virtuels (VPN) et confidentialité

Un réseau privé virtuel (ou VPN) est un système permettant de simuler une connexion LAN entre plusieurs machines, ou plusieurs sites d'entreprise, en les faisant transiter par un réseau non local (par exemple un WAN comme Internet).

De cette façon, les machines peuvent communiquer de façon sécurisée (par chiffrement via SSL et authentification par clefs publiques/privées) entre elles, comme si elles étaient en réseau local. En effet, les adresses IP utilisées pour communiquer font partis d'un adressage propre au VPN/LAN.

Dans le cadre de ce TP, nous avons relié deux machines entre elles, en créant un « Bridge Ethernet » avec OpenVPN.

Pour installer OpenVPN sous Debian, nous avons utilisé la commande :

```
# apt-get install openvpn
```

Sur la machine serveur, on se place dans le dossier `/etc/openvpn` et on exécute la commande suivante pour générer une clef statique :

```
# openvpn --genkey --secret static.key
```

Ensuit, on échangé la clef du serveur vers le client, par ssh par exemple :

```
# scp static.key toto@172.17.4.4:/home/toto/
```

Du côté du serveur OpenVPN, nous avons modifié le fichier de configuration (openvpn.conf) comme suit:

```
dev tun
ifconfig 192.168.0.1 192.168.0.2
secret static.key
```

Du côté du client OpenVPN, nous avons déplacé la clef statique dans le dossier /etc/openvpn puis nous avons configuré le fichier de configuration openvpn comme suit :

```
remote remotedomain
dev tun
ifconfig 192.168.0.2 192.168.0.1
secret static.key
```

Enfin, nous avons pu valider le fonctionnement à l'aide d'un message ICMP ECHO REQUEST (en le réactivant dans le noyau et en l'autorisant dans le firewall) :

```
pc0b004-3:/etc/openvpn# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.097 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.079 ms

--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.079/0.088/0.097/0.009 ms
```

Ce type de configuration est idéal pour créer une connexion point à point, mais présente l'inconvénient du partage de la clef, qui la rend à priori moins sure. Cependant, pour relier plusieurs réseaux d'entreprise entre eux, ou plusieurs sites, ceci peut être une bonne solution.

Conclusion

Ces séances de travaux pratiques nous ont appris à mettre en œuvre une stratégie sécuritaire de déploiement d'un serveur destiné à un réseau de production en entreprise.

A titre personnel, je pense que ces séances ont été très utiles. En effet, j'administre de façon personnelle plusieurs serveurs Unix en production, et suis en contact étroit avec le pôle Administration Systèmes et Réseaux de l'entreprise pour laquelle je travaille.

Mon constat est qu'il est assez effarant de constater à quel point les règles citées dans ce document sont peu connues ou respectées, même par des professionnels du domaine : compilateurs installés sur des machines en production, pas d'audit de sécurité, absence d'IDS, impossibilité de détecter les intrusions ou tentatives...

De façon plus générale encore, de nombreuses attaques réalisées par des hackers sont faites à partir d'ordinateurs eux-mêmes hackés : de nombreux utilisateurs installent des systèmes Unix en production sans savoir les administrer, ce qui les rend d'autant plus vulnérable.

En d'autres termes, bien qu'initié dans le domaine de la sécurité des systèmes et réseaux, les recommandations fournies durant le cours et durant les séances de TP par Sébastien Lelong et Yves Lavanant m'ont été très utiles pour améliorer la sécurité de mes systèmes Linux.