

Mise en place d'un réseau sécurisé - R5

Abdalla ALTUNAJI, Hugo ETIEVANT, Remy FABREGES,
Benoît MAYNARD, Jean-François RODRIGUEZ, Yohan VALETTE

Maîtrise IUP Génie Informatique Réseau
Université Claude Bernard - Lyon 1

Novembre 2002

Chapitre 1

Introduction

Ce rapport a pour but de présenter la construction pas à pas d'un réseau sécurisé. Elle sera suivie d'une phase de tests et d'attaques à l'aide d'outils logiciels disponibles sur Internet.

1.1 Sujet

Création d'un réseau informatique sécurisé. Ce réseau comprendra un routeur utilisant un Firewall, un ou des clients dans un réseau local, un bastion proposant au moins un service web intranet. Un service Telnet distant ou SSH permettra de communiquer avec un client distant.

Protection du réseau local. Les clients locaux devront être protégés du monde extérieur.

VPN. Un tunnel sécurisé devra permettre à un client extérieur de se connecter de façon sûre au routeur.

Tests. La sécurité de l'ensemble devra être dûment testée par les outils adéquats.

1.2 Limitations

Par manque de matériel, nous n'avons pas pu installer de DMZ (zone démilitarisée) puisque nous n'avons à notre disposition que deux interfaces réseaux sur notre passerelle alors que trois nous auraient été nécessaires (client, extérieur, bastion)

L'absence d'accès à Internet depuis le réseau «global» des salles de TP, nous a considérablement freiné. La phase de recherche et de téléchargement des applications qui nous étaient nécessaires devait se faire en dehors des heures de TP.

Par ailleurs, une panne sur une machine nous a privé d'un client.

Le problème récurrent du partage des salles de TP avec d'autres élèves implique :

- la perte systématique de nos scripts,
- la configuration du système renouvelée à chaque séance,
- la ré-installation des applications.

1.3 Répartition des tâches

Nous avons réparti les ressources de notre équipe en fonction des besoins de notre réseau : firewall, scanning, attaques externes, attaques internes et VPN. Pour voir la répartition de tâches, voir le tableau 1.1 page 4.

Nom	Tâche	Outils
Altunaiji	installation d'un VPN sur SSH	
Etiévant	installation firewall, tests Telnet	ipchains
Fabrèges	scanning, piratage	nmap
Maynard	client distant	ipchains
Rodriguez	attaques et tests	nessus, uito
Valette	crack mot de passe, intranet	crack, apache

TAB. 1.1 – Répartition des tâches

1.4 Présentation orale

L'équipe présentera un exposé oral le vendredi 13 décembre pendant 1 heure. Avec comme support un document Power Point.

Chapitre 2

Architecture

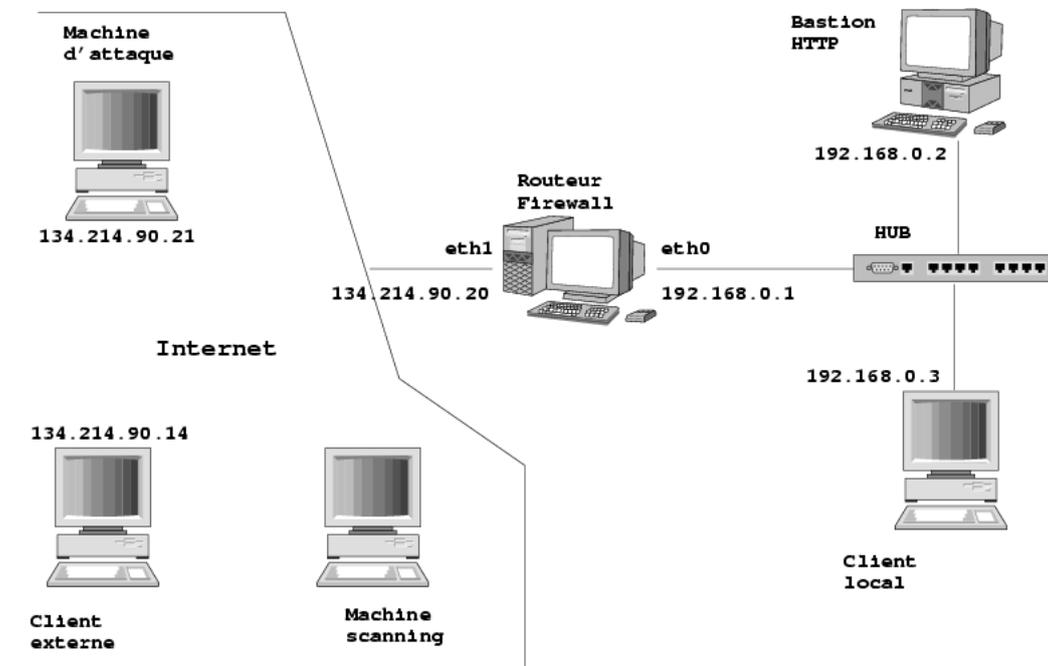


FIG. 2.1 – Schéma de notre réseau

Chapitre 3

Firewall

L'installation d'un firewall (pare feu) sur le routeur qui sert de passerelle entre notre réseau local et le reste du monde est une étape obligée de la sécurisation de nos installations.

- Les noyaux 2.0 disposent de `IPfwadm`
- Les noyaux 2.2 ont `IPchains` et `IPMasqadm`
- Les noyaux 2.4 possèdent `IPtables` avec `Netfilter`

```
$ uname -sr  
Linux 2.2.20
```

Notre routeur tournant sous Linux de version de noyau 2.2, nous avons choisi d'utiliser l'outil natif de Linux : *Ipchains*.

3.1 L'outil ipchains

3.1.1 Présentation

Le firewall ipchains permet d'établir des règles de filtrage organisées selon une hiérarchie de chaînes arborescente. Il ne sera pas discuté ici de la syntaxe de cet outil mais de la stratégie de sécurisation de notre passerelle.

3.1.2 Fonctionnement

Les règles de filtrage déterminent le devenir des paquets grâce à une police qui peut être l'une de celles du tableau 3.2 page 10 ou bien une redirection vers une autre chaîne. Les règles sont groupées en chaînes qui peuvent être celles de base (voir le tableau 3.1 page 10) ou bien d'autres définies par l'utilisateur. Les paquets se présentant au pare-feu arrivent automatiquement dans la chaîne `input`. Ceux qui sont émis par le pare-feu passent dans la chaîne `output` avant leur sortie physique effective. Un paquet qui entre dans une chaîne teste toutes les règles de la chaîne jusqu'à en trouver une qui lui corresponde. Et il obéit à la police spécifiée par la règle trouvée. Si aucune règle s'appliquant au paquet n'a été trouvée, alors c'est la police par défaut de la chaîne qui fait loi. Il est donc important, avant de créer les règles, de définir la police par défaut d'une chaîne. Par sécurité on rejette en entrée (`input REJECT`) et en sortie (`output REJECT`) et on ignore en redirection (`forward DENY`).

3.1.3 Critères de filtrage

Les règles de filtrage reposent sur des critères de sélection très variés :

- adresse IP source
- adresse IP destination
- port source
- port destination
- protocole (UDP, TCP, ICMP)
- interface
- TOS (qualité de service)
- drapeaux spéciaux du protocole TCP
- types de codes spéciaux du protocole ICMP

3.1.4 Actions possibles sur les paquets

Les règles peuvent — en plus de la police — définir différentes actions sur un paquet :

- mise en fichier de log
- redirection vers une autre chaîne
- marquage d'un paquet
- application d'une police

3.1.5 Mode de construction des chaînes et des règles

Les options de *ipchains* pour la gestion des chaînes et des règles sont les suivantes :

- Chaînes
 - création N
 - vidage F
 - suppression X
 - affectation d'une police P
 - affichage L
- Règles
 - ajout A
 - insertion ordonnée I
 - suppression D
 - remplacement R
 - changement d'ordre R
 - test C

3.2 Stratégie

Un script de démarrage permet d'appliquer notre politique de sécurité :

- IP masquerading
- Telnet limité à un client déterminé
- Interdiction des ping
- Palier à l'IP Spoofing
- Autoriser le trafic web du réseau privé vers l'extérieur
- Limiter les autres services (FTP, rlogin, finger...)
- Conserver le trafic sensible dans un fichier de log (`/etc/log/messages`)

Ce script rend portable et réutilisable notre configuration.

3.2.1 Réseau local invisible

Nous possédons un réseau local de machines qui doivent être invisibles depuis l'extérieur. Mais elles doivent pouvoir accéder au service web à l'extérieur. Pour cela, on va mettre en place du *Masquerading*.

Le masquerading¹ est directement pris en charge par Linux équipé d'un noyau de version supérieur à 2.2.x. Notre distribution de Debian ne pose aucun problème de ce côté-là.

La passerelle va assurer le partage de la connexion réseau vers l'extérieur. Elle possède deux interfaces réseau, l'une connectée au réseau privé, l'autre à Internet. Incidemment, notre passerelle va posséder deux adresses IP : l'une publique fournie par notre fournisseur d'accès, et l'autre privée que l'on va choisir à notre convenance parmi les intervalles de valeurs suivants : 10.0.0.0/24, 172.16.0.0, 192.168.0.0/24. Nous choisirons d'utiliser 192.168.0.0/24. Pour le schéma représentant la translation d'adresse, voir la figure 3.1 page 9

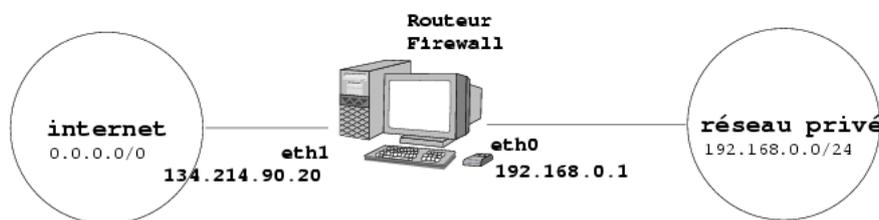


FIG. 3.1 – Schéma de l'IP masquerading

Notre script de sécurisation comprend les éléments suivants :

- Activation du forwarding automatique (par MAJ d'une valeur système du noyau) :
`echo 1 > /proc/sys/net/ipv/ip_forward`
- Filtrage de la chaîne *forward* par application de la police *DENY* :
`ipchains -P forward DENY`
- Définition du masque d'adresses du réseau privé sur l'interface *eth0* :
`ifconfig eth0 up 192.168.0.1 netmask 255.255.255.0`
- Définition de l'adresse IP publique sur l'interface *eth1* connecté à Internet :
`ifconfig eth1 up 134.214.90.20 netmask 255.255.252.0`
- Mise en route de l'IP masquerading bidirectionnel :
`ipchains -b -A forward -s 192.168.0.0/24 -j MASQ`

Syntaxe de configuration d'une carte réseau :

```
ifconfig interface [ up | down ] adresse [ netmask masque ]
```

3.3 Passerelle sécurisée

3.3.1 Initialisation des règles de filtrage

Par défaut, on commence par rejeter tous les paquets quels que soient le protocole, la source, le port... On applique la police *REJECT* aux chaînes de base *input* et *output* qui rejette les paquets en entrée et en sortie par l'envoi de paquet ICMP (car on est poli). La police *DENY* sur la chaîne *forward* rend invisible le réseau local en interdisant la redirection de tout paquet tout en interdisant l'envoi de message d'erreur. Et enfin suppression de toutes les règles existantes.

Syntaxe d'application d'une police à une chaîne :

```
ipchains -P chaîne police
```

¹L'IP masquerading ou *traduction d'adresses IP* s'appelle en fait NAT pour "Network Address Translation"

Syntaxe de suppression de toutes les règles :

```
ipchains -F [ police ]
ipchains -P input REJECT
ipchains -P output REJECT
ipchains -P forward DENY
ipchains -F
```

Chaîne	Description
input	traite les paquets d'entrée
output	traite les paquets de sortie
forward	traite les paquets de redirection

TAB. 3.1 – Chaînes de base

Police	Description
ACCEPT	laisse passer les paquets
REJECT	ne laisse pas passer les paquets, mais retourne un message d'erreur ICMP
DENY	ne laisse pas passer les paquets en mode silencieux
MASQ	utilise l'IP masquering pour le rediriger

TAB. 3.2 – Liste des polices utilisées

3.3.2 Accès http

On souhaite autoriser les machines locales à se connecter aux sites web extérieurs. Pour cela on laisse passer tout datagramme TCP sortant vers une destination de port 80. On laisse entrer tout datagramme TCP provenant d'un port 80 sauf les SYN ².

```
ipchains -A forward -p tcp -s 192.168.0.0/24 -d 0.0.0.0/0 80 -j MASQ
ipchains -A forward -p tcp ! -y -s 0.0.0.0/0 80 -d 192.168.0.0/24 -j MASQ
```

Dans nos tests, on se connecte avec succès au serveur web <http://134.214.90.19/> de Philippe CANNETTE.

3.3.3 Processus locaux

On va autoriser les processus locaux à accéder comme bon leur semble à l'interface `loopback`.

```
ipchains -A input -i lo -j ACCEPT
ipchains -A output -i lo -j ACCEPT
```

3.3.4 Réseau privé

On part du principe que les machines de notre réseau privé sont sûres. On va donc les autoriser à communiquer avec la passerelle sans restriction.

```
ipchains -A input -s 192.168.0.0/24 -j ACCEPT
ipchains -A output -d 192.168.0.0/24 -j ACCEPT
```

²Les demandes de connexions TCP entrantes sont formulées par une requête SYN.

3.3.5 anti - IP Spoofing

L'IP Spoofing consiste à usurper l'identité de quelqu'un. Dans notre cas, on prévient le cas où de l'extérieur, une machine voudrait se faire passer pour appartenant à notre réseau privé. Notre salut vient alors du filtrage sur l'interface.

Nous gérons également le cas où l'on veuille se faire passer pour... nous-même !

On va créer une chaîne utilisateur afin de gérer l'IP Spoofing qu'il faut interdire mais aussi mettre en fichier de log.

```
ipchains -N ipspooft
ipchains -F ipspooft
ipchains -A ipspooft -l -j DENY
ipchains -A input -i eth1 -s 192.168.0.0/24 -j ipspooft
ipchains -A input -i eth1 -s 134.214.90.20 -j ipspooft
```

3.3.6 Restrictions telnet

Pour des besoins de maintenance à distance, on va autoriser une machine à accéder à la passerelle avec Telnet. Cette machine a pour adresse : 134.214.90.14. Le fichier `/etc/services` nous apprend que telnet utilise le port 23.

```
# Telnet en entree
ipchains -N tel-in
ipchains -F tel-in
ipchains -A tel-in -s 134.214.90.14 -d 134.214.90.20 -j ACCEPT
ipchains -A tel-in -l -j REJECT
ipchains -A input -i eth1 -p tcp -s 0.0.0.0/0 23 -j tel-in

# Telnet en sortie
ipchains -N tel-out
ipchains -F tel-out
ipchains -A tel-out -s 134.214.90.20 -d 134.214.90.14 -j ACCEPT
ipchains -A tel-out -l -p tcp ! -y -j REJECT
ipchains -A output -i eth1 -p tcp -s 0.0.0.0/0 23 -j tel-out
```

3.3.7 Trafic IP

On autorise les connexions entrantes/sortantes IP.

```
ipchains -A output -p tcp -i eth1 -j ACCEPT
ipchains -A input -p tcp -i eth1 -j ACCEPT
```

3.3.8 Trafic ICMP

Il existe deux outils : `ping` et `traceroute` très utilisés par les pirates car leur permettant d'obtenir des informations sur une machine connectée sur Internet et de préparer une attaque. Ils reposent tous deux sur le protocole ICMP.

Internet Control Message Protocol Le protocole ICMP est défini par la RFC 950, c'est un mécanisme de contrôle des erreurs au niveau IP. Dans le cas où un paquet IP ne peut être routé, délivré ou acheminé correctement sur le réseau où encore que la QoS (Quality Of Service) ne peut être respectée, un message d'erreur est envoyé à l'émetteur sous la forme d'un message ICMP.

Le message ICMP est encapsulé dans l'IP. L'entête du message ICMP est composé de trois champs : le **type** (qui contient le code d'erreur), le **code** (facultatif, complète l'information précédente) et le **checksum** (qui porte la somme de vérification de l'intégrité de l'en-tête ICMP).

Il y a génération d'un message ICMP dans les cas suivants :

- demande et réponse d'écho,
- destination inaccessible,
- expiration de délai pour un datagramme IP (champs TTL à 0),
- limitation du débit de la source,
- redirection (changement de route),
- problème de paramètre avec un datagramme,
- demande et réponse de temps,
- demande et réponse de masque de sous-réseau.

ping C'est un outil basé sur le protocole ICMP qui permet de déterminer si une machine est active sur l'Internet. Il envoie un message ICMP de type **ECHO_REQUEST** à une machine qui elle-même est tenue de répondre par un message ICMP de type **ECHO_REPLY**. Le ping peut aussi servir à tester la connexion avec la machine distante en envoyant une requête toutes les secondes et en produisant des statistiques sur les temps de réponse et le nombre de paquets perdus.

traceroute C'est un outil qui permet de suivre le trajet d'un paquet jusqu'à la machine destination. Pour cela, il envoie un message ICMP avec un TTL égal à 1, puis il recommence en incrémentant le TTL de 1 à chaque envoi. A chaque fois que le TTL arrive à 0, le routeur renvoie un message ICMP d'erreur comportant en données son adresse. On peut donc connaître la route exacte empruntée.

Nous allons protéger notre réseau du trafic ICMP indésirable provenant de l'extérieur afin de garder invisible notre réseau privé, éviter le «ping de la mort»³ et autres attaques.

```
# ICMP en entree
ipchains -N icmp-in
ipchains -F icmp-in
ipchains -A icmp-in -l -p icmp --icmp-type echo-request -j DENY
ipchains -A input -i eth1 -p icmp -j icmp-in
```

```
# ICMP en sortie
ipchains -N icmp-out
ipchains -F icmp-out
ipchains -A icmp-out -l -p icmp --icmp-type echo-reply -j DENY
ipchains -A output -i eth1 -p icmp -j icmp-out
```

3.3.9 Fichier de log

Voici une ligne extraite de notre fichier de log `/etc/log/messages` dont les champs[2] en sont expliqués dans le tableau B.1 page 67 :

```
Nov  8 10:25:53 b710pbv kernel: Packet log:
```

³Le ping de la mort (Ping Of Death) consiste en l'envoi de paquets ICMP illégalement grands (> 64 Ko) qui font déborder les buffers de la pile TCP de la carte ethernet du récepteur une fois les fragments réassemblés. La solution est de bloquer les fragments ICMP. Sachant que les paquets ICMP normaux ne sont pas assez gros pour nécessiter la fragmentation, on peut bloquer le ping de la mort sans pour autant nécessairement bloquer les ping normaux. Ses conséquences peuvent aller du simple gel des processus en cours jusqu'à un redémarrage complet de la machine.

```
input REJECT eth1 PROTO=17 134.214.90.16:138 134.214.91.255:138
L=236 S=0x00 I=51490 F=0x0000 T=128 (#7)
```

Donc, le 8 novembre à 10h25m et 53s, la machine `b710pbv` a mis en log via son `kernel` un paquet provenant de la machine `134.214.90.16` (port `138`) via l'interface `eth1` à destination de `134.214.91.255` (port `138`). Ce paquet est passé par la chaîne de base `input` et a subi la police `REJECT`, c'est-à-dire qu'il a été poliment rejeté. Le protocole utilisé était le numéro `138`. Le règle `ipchains` qui a mis ce paquet en log était la numéro `7`. Son `TTL` vaut `128`. Le `TOS` est à `Normal_Service`. La taille du paquet est de `236` octets. Le port utilisé est connu sous la dénomination `netbios-dgm` (NETBIOS Datagram Service)⁴. L'identifiant du paquet IP est `51490`.

3.4 Testons le service ftp

Lors d'une session FTP depuis une machine distante vers notre passerelle, et si on tente de télécharger le fichier de mots de passe :

```
ftp> get /etc/passwd
200 PORT command successful.
550 /etc/passwd is marked unretrievable
```

et ce, à cause de la directive `noretrieve` dans `ftppaccess`.

Le fichier de log `/etc/log/messages` nous livre alors ceci :

```
Nov  8 12:00:25 b710pbv wu-ftpd[11717]: FTP LOGIN FROM 134.214.90.21 [134.214.90.21], jeff
Nov  8 12:00:36 b710pbv wu-ftpd[11717]: jeff of 134.214.90.21 [134.214.90.21]
                                                tried to download /etc/passwd
Nov  8 12:00:42 b710pbv wu-ftpd[11717]: FTP session closed
```

On peut y lire que quelqu'un s'est connecté depuis la machine `134.214.90.21` à midi le 8 novembre via le protocole `ftp` sous le nom d'utilisateur `jeff`. Il a essayé de télécharger le fichier de mots de passe `/etc/passwd` sans succès, puis s'est déconnecté peu après !

3.5 Testons le service telnet

Des tests de sécurité sur le protocole Telnet ont été effectués. Ils révèlent quelques surprises.

Le fichier `/etc/services` révèle que le protocole Telnet utilise le port `23`.

3.5.1 Connexion

Le connexion directe avec le login `root` est interdite car il existe un fichier des logins prohibés.

Avant toute chose, on procède à la création d'un nouvel utilisateur `jeff` avec la commande `adduser` sur la passerelle `134.214.90.20`, puis connexion via la commande `telnet` à partir du client externe `134.214.90.14`.

```
$ telnet 134.214.90.20
Trying 134.214.90.20...
Connected to 134.214.90.20.
Escape character is '^]'
```

⁴Protocole de gestion des noms d'ordinateurs locaux et d'ordinateurs distants de réseaux locaux développés par les systèmes d'exploitation Microsoft. Pour permettre aux machines sous Windows de faire tourner leurs applications sur de grands réseaux, Microsoft a développé NetBios sur TCP/IP. Le Netbios reste donc le standard de fait des systèmes d'exploitation Microsoft, mais il utilise les couches TCP/IP pour le transport de données inter-réseau.

```
Debian GNU/Linux 2.2 134.214.90.20
miage login: jeff
Password:
Last login: Wed Nov 13 14:48:32 2002 from 134.214.90.14 on ttyp2
Linux 2.2.20 #2 Sun Nov 4 10:45:25 CET 2001 i686 unknown
```

Most of the programs included with the Debian GNU/Linux system are freely redistributable; the exact distribution terms for each program are described in the individual files in `/usr/doc/*/copyright`

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.
No mail.

Usage: mesg [y|n]

```
$ echo $HOME
/home/jeff
```

3.5.2 Navigation

On se trouve alors dans le répertoire de connexion de l'utilisateur `jeff` : `/home/jeff` avec les droits en lecture et écriture sur ce répertoire. On navigue dans l'arborescence des répertoires de `/home` et on réussit à supprimer des répertoires créés par `root` ! Ces répertoires n'étaient affectés à aucun utilisateur mais créés par le super utilisateur. Il est étonnant que ces répertoires aient ainsi pu être supprimés par un utilisateur normal. Cela doit probablement venir du masque de fichier...

En navigant dans les répertoires des autres utilisateurs, sauf modification des permissions accordées aux autres utilisateurs du groupe et des autres par le propriétaire, la lecture est autorisée mais pas l'écriture.

3.5.3 Accès aux périphériques

Testons l'accès aux périphériques : lecteur de disquette et lecteur de CD-ROM.

```
mduir a:
touch foo.bar
mcopy foo.bar a:
```

L'accès au lecteur de disquette en lecture et en écriture via l'utilisation des MTools est donc permise.

```
mount /floppy
ls /floppy
cp foo.bar /floppy
umount /floppy
```

Le montage, l'accès en lecture/écriture et le démontage sont autorisés sur le lecteur de disquette `/floppy`.

```
mount /cdrom
ls /cdrom
umount /cdrom
```

Le montage, l'accès en lecture et le démontage sont autorisés sur le lecteur de CD-ROM `/cdrom`.

En fait, les permissions d'accès sur les fichiers d'un périphérique sont accordées en fonctions des particularités intrinsèques du périphérique (pas d'écriture sur `/cdrom`) mais aussi en terme de droits maximums uniquement pour l'utilisateur qui le monte.

3.5.4 Droits super utilisateur

```
$ su root
Password:
$ echo "Bonjour" > /root/toto.txt
$ exit
exit
```

Une fois connecté à la passerelle sous l'identité d'un utilisateur lambda (ici : `jeff`), il est toujours possible de changer d'identité grâce à la commande `su`. Et ainsi d'acquérir les droits `root` si l'on connaît par avance son mot de passe. Après avoir acquis les droits `root`, on peut écrire dans son répertoire.

3.5.5 Déconnexion

Et fermeture de la connexion via la commande `exit` .

```
$ exit
exit
Connection closed by foreign host.
```

3.6 SSH

Le protocole Telnet se révèle très pratique pour effectuer une connexion distante par émulation de terminal, mais n'est pas sécurisé puisque le login et le mot de passe transitent en clair sur le réseau. Dans le cas d'une connexion entre deux machines de notre réseau privé, ce risque peut être accepté ; mais pas depuis une machine du réseau public. Car une sonde⁵ ou un sniffer⁶ peuvent avoir été installés sur les éléments intermédiaires du réseau Internet.

Pour sécuriser les transactions, on utilise le protocole SSH (Secure SHell) qui repose sur le cryptage des données échangées. Le cryptage/décryptage s'effectue au moyen de paires de clefs publique/privée. Ainsi le login et le mot de passe restent cachés.

Pour résumer SSH permet d'authentifier fortement les deux bouts de la connexion, de rediriger dans un tunnel chiffré tous les flux entre les deux machines (login, mot de passe, fichiers...) voire même de les compresser en live de façon transparente...

⁵Une sonde est un matériel permettant d'intercepter tout le trafic circulant sur le lien sur lequel elle est branchée. Elle reste invisible car passive.

⁶Un sniffer est un logiciel qui conserve la trace du trafic.

Chapitre 4

SSH

4.1 Introduction

SSH permet aux utilisateurs de se connecter à distance à un autre système hôte. Contrairement à rlogin ou telnet, SSH chiffre la session de connexion et empêche ainsi aux hackers de détecter les données sensibles qui peuvent transiter. En utilisant des méthodes sécurisées pour vous connecter à distance à d'autres systèmes, vous réduisez les risques en matière de sécurité, pour votre système et le système distant.

4.2 Pourquoi utiliser SSH ?

L'interception de paquets, la mystification DNS et l'IP spoofing ainsi que la diffusion de fausses informations de routage ne sont que quelques exemples des menaces qui planent lors des communications en réseau.

L'utilisation du protocole SSH pour effectuer une connexion shell à distance ou copier des fichiers permet de faire diminuer sensiblement ces menaces à la sécurité. La signature numérique d'un serveur fournit la vérification pour son identité. En outre, la communication complète entre un système client et un système serveur ne peut être utilisée si elle est interceptée car tous les paquets sont chiffrés. De plus, il n'est pas possible d'usurper l'identité d'un des deux systèmes, parce que les paquets sont chiffrés et leurs clés ne sont connues que par les systèmes local et distant.

4.3 Introduction à la cryptographie

Le chiffrement se fait généralement à l'aide d'une clé de chiffrement, le déchiffrement nécessite quant à lui une clé de déchiffrement. On distingue généralement deux types de clés :

- Les clés symétriques : il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- Les clés asymétriques : il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, deux clés différentes sont utilisées — l'une pour le chiffrement et l'autre pour le déchiffrement

4.3.1 Le cryptage symétrique :

Le cryptage symétrique est basé sur une clé partagée entre les deux parties communicantes. Cette même clé sert à crypter et décrypter les messages. L'avantage de la cryptographie symétrique est

sa rapidité d'exécution car elle met en oeuvre des opérations simples. Mais, le principal problème est le partage de la clé : Comment une clé utilisée pour sécuriser peut être transmise sur un réseau non-sécurisé ? La difficulté engendrée par la génération, le stockage et la transmission des clés limite l'utilisation des clés symétriques surtout sur Internet. On appelle l'ensemble de ces trois processus la gestion des clés : « key management ».

Les algorithmes informatiques développés pour réaliser des opérations de cryptographie symétriques sont :

1. DES (Data Encryption Standard, 1974, inventé par la NSA, encore appelé DEA (ANSI) ou DEA-1 (ISO)),
2. Triple DES ou DES3 (1985),
3. RC2, RC4, RC5 et RC6 (1987, RSA Data Security, Inc),
4. IDEA (International Data Encryption Algorithm, 1990-1992),
5. AES (Advanced Encryption Standard, 1997 , commercialisation 2001).

La force d'un algorithme symétrique est proportionnelle à la taille de sa clé. Plus cette taille est grande, plus le nombre de combinaisons à tester pour trouver la clé et décrypter les données sera élevé. Plus le nombre de clés nécessaires est élevé, plus il sera difficile de déchiffrer l'algorithme.

Pour résoudre ces problèmes de transmission de clés, les mathématiciens ont inventé le cryptage asymétrique qui utilise une clé privée et une clé publique.

4.3.2 Le cryptage asymétrique :

Ce système de cryptage utilise deux clés différentes pour chaque utilisateur : une est privée et n'est connue que de l'utilisateur ; l'autre est publique et donc accessible par tout le monde.

Les clés publiques et privées sont mathématiquement liées par l'algorithme de cryptage de telle manière qu'un message crypté avec une clé publique ne puisse être décrypté qu'avec la clé privée correspondante. Une clé est donc utilisée pour le cryptage et l'autre pour le décryptage. Ce cryptage présente l'avantage de permettre le placement de signatures numériques dans le message et ainsi permettre l'authentification de l'émetteur.

Le principal avantage de ce cryptage est de résoudre le problème de l'envoi de clé symétrique sur un réseau non sécurisé. Bien que plus lent que la plupart des cryptages à clé symétrique, il reste préférable à l'utilisation critique. En pratique il est utilisé pour :

- L'échange d'une clé symétrique,
- La signature d'un hachage d'un message.

Les trois algorithmes à clé publique suivants sont les plus fréquemment employés :

1. RSA : (Rivest-Shamir-Adleman,1978) : RSA est unique parmi les algorithmes à clé publique utilisés, en ce sens qu'il peut effectuer des opérations de signature numérique et d'échange de clés.
2. DSA : (Digital Signature Algorithm) : DSA tire sa sécurité de la difficulté du calcul de logarithmes discrets. Cet algorithme peut uniquement être utilisé pour les opérations de signature numérique (pas pour le cryptage de données).
3. Diffie-Hellman : La sécurité de Diffie-Hellman est liée à la difficulté du calcul des logarithmes discrets dans un champ fini. L'algorithme de Diffie-Hellman peut uniquement être utilisé pour l'échange de clés.

Ces algorithmes utilisent des fonctions mathématiques très complexes, par conséquent, ils sont beaucoup plus lents que les algorithmes à clé privée.

La différence fondamentale entre les clés symétriques et asymétriques provient de leur utilisation :

- La clé symétrique sert à crypter des grands volumes de données,
- La clé asymétrique sert à crypter la clé symétrique et à la transporter en sécurité.

4.4 Scénario d'usage

Lorsqu'un client communique avec un serveur au moyen d'un protocole SSH, de nombreux éléments importants sont négociés afin que les deux systèmes puissent créer correctement la couche transport. Les opérations ci-dessous ont lieu durant cet échange :

- l'échange des clés
- l'algorithme de clé publique à utiliser
- l'algorithme de chiffrement symétrique à utiliser
- l'algorithme d'authentification de message à utiliser
- l'algorithme repère (hash) à utiliser

Durant l'échange des clés, le serveur s'identifie au client au moyen d'une clé hôte. Evidemment, si le client communique pour la première fois avec ce serveur, la clé du serveur lui est inconnue. OpenSSH contourne ce problème en permettant au client d'accepter la clé hôte du serveur lors de leur première connexion SSH. Ensuite, lors des connexions suivantes, la clé hôte du serveur peut être vérifiée au moyen d'une version enregistrée sur le client, ce qui permet au client de s'assurer qu'il communique bien avec le serveur désiré.

Attention : Un hacker pourrait se faire passer pour le serveur SSH lors de la première connexion car le système local ne reconnaît pas le serveur désiré d'un autre serveur. Afin d'éviter cela, contrôlez l'intégrité d'un nouveau serveur SSH en contactant l'administrateur du serveur avant d'établir la première connexion.

4.5 Exiger SSH pour les connexions à distance

Afin que le protocole SSH soit vraiment efficace et protège vos connexions réseau, vous devez absolument cesser d'utiliser des protocoles de connexion non sécurisés, tels que telnet et rsh. Autrement, le mot de passe d'un utilisateur pourrait être protégé au moyen de ssh, mais être capté lors d'une connexion ultérieure de ce même utilisateur au moyen de telnet.

Ci-dessous quelques services que vous devez désactiver :

- telnet
- rsh
- ftp
- rlogin
- wu-ftpd
- vsftpd

4.6 Sources

Les sources sont disponibles aux adresses :

- OpenSSH : <http://www.OpenSSH.com>
- OpenBSD : <http://www.openbsd.org/openssh>
- FreeSSH : <http://www.freessh.org>
- OpenSSL : <http://www.openssl.org>

Notez également que les paquetages OpenSSH requièrent le paquetage OpenSSL (openssl). OpenSSL installe de nombreuses bibliothèques cryptographiques importantes qui aident OpenSSH à chiffrer les communications.

4.7 Fichiers de configuration d'OpenSSH

OpenSSH est constitué de deux ensembles de fichiers de configuration, un pour les programmes clients (ssh) et l'autre pour le serveur (sshd).

Les informations de configuration SSH qui s'appliquent à l'ensemble du système sont stockées dans le répertoire `/etc/ssh` :

- `ssh_config` - fichier de configuration client SSH pour l'ensemble du système.
- `sshd_config` - fichier de configuration pour le daemon sshd du serveur.
- `ssh_host_dsa_key` - clé DSA privée utilisée par sshd.
- `ssh_host_dsa_key.pub` - clé DSA publique utilisée par sshd.
- `ssh_host_rsa_key` - clé RSA privée utilisée par sshd .
- `ssh_host_rsa_key.pub` - clé RSA publique utilisée par sshd SSH.

Les informations de configuration SSH spécifiques à l'utilisateur sont stockées dans son répertoire personnel à l'intérieur du répertoire `/.ssh/` :

- `authorized_keys` - ce fichier contient une liste de clés publiques "autorisées". Si un utilisateur se connecte et prouve qu'il connaît la clé privée correspondant à l'une de ces clés, il obtient l'authentification.
- `known_hosts` - Ce fichier contient les clés hôte DSA des serveurs SSH auxquels l'utilisateur s'est connecté. Ce fichier est très important car il garantit que le client SSH se connecte au bon serveur SSH.

4.7.1 Le fichier `ssh_config` :

```
#      $OpenBSD: ssh_config,v 1.16 2002/07/03 14:21:05 markus Exp $

# This is the ssh client system-wide configuration file.  See
# ssh_config(5) for more information.  This file provides defaults for
# users, and the values can be changed in per-user configuration files
# or on the command line.

# Configuration data is parsed as follows:
# 1. command line options
# 2. user-specific file
# 3. system-wide file
# Any configuration value is only changed the first time it is set.
# Thus, host-specific definitions should be at the beginning of the
# configuration file, and defaults at the end.

# Site-wide defaults for various options

# Host *
#   ForwardAgent no
#   ForwardX11 no
```

```
# RhostsAuthentication no
# RhostsRSAAuthentication no
# RSAAuthentication yes
# PasswordAuthentication yes
# HostbasedAuthentication no
# BatchMode no
# CheckHostIP yes
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/identity
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# Port 22
# Protocol 2,1
# Cipher 3des
# Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc
# EscapeChar ~
```

4.7.2 Le fichier sshd_config :

```
# $OpenBSD: sshd_config,v 1.59 2002/09/25 11:17:16 markus Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options change a
# default value.

#Port 22
#Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::

# HostKey for protocol version 1
#HostKey /usr/local/etc/ssh_host_key
# HostKeys for protocol version 2
#HostKey /usr/local/etc/ssh_host_rsa_key
#HostKey /usr/local/etc/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 3600
#ServerKeyBits 768

# Logging
```

```
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 120
#PermitRootLogin yes
#StrictModes yes

#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile      .ssh/authorized_keys

# rhosts authentication should not be used
#RhostsAuthentication no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
# For this to work you will also need host keys in /usr/local/etc/ssh_known_hosts
#RhostsRSAAuthentication no
# similar for protocol version 2
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
#IgnoreUserKnownHosts no

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

#AFSTokenPassing no

# Kerberos TGT Passing only works with the AFS kaserver
#KerberosTgtPassing no

# Set this to 'yes' to enable PAM keyboard-interactive authentication
# Warning: enabling this may bypass the setting of 'PasswordAuthentication'
#PAMAuthenticationViaKbdInt no
```

```
#X11Forwarding no
#X11DisplayOffset 10
#X11UseLocalhost yes
#PrintMotd yes
#PrintLastLog yes
#KeepAlive yes
#UseLogin no
#UsePrivilegeSeparation yes
#PermitUserEnvironment no
#Compression yes

#MaxStartups 10
# no default banner path
#Banner /some/path
#VerifyReverseMapping no

# override default of no subsystems
Subsystem sftp /usr/local/libexec/sftp-server
```

4.7.3 Un fichier très simple de sshd_config :

```
#sshd_config file by Abdalla
AllowGroups vpn
AllowUsers vpn
HostKey /etc/ssh/identity
```

4.8 Installation

Il faut installer les paquetages Openssl, ensuite les paquetages OpenSSH qui est constitué de deux ensembles de fichiers de configuration, un pour les programmes clients (ssh) et l'autre pour le serveur (sshd).

4.9 Configuration

Une fois l'installation de différents paquetages est effectuée sur les deux machines, on commence par le client :

4.9.1 Client

Créer un groupe *vpn* et un client *vpn*, et se connecter en tant qu'utilisateur *vpn*

On lance la génération de deux clés par la commande : *ssh-keygen*

```
vpn@b710pbw:~$ ssh-keygen
Generating RSA keys: .....oooo0..oooo0
Key generation complete.
Enter file in which to save the key (/home/vpn/.ssh/identity): key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in key.
Your public key has been saved in key.pub.
The key fingerprint is:
1024 0d:83:b1:6d:89:05:70:10:ad:69:b5:aa:c7:2c:dc:76 vpn@b710pbw
```

Le résultat de cette commande est deux fichiers dont un contient la clé privée et l'autre contient la clé publique. Les clés privées doivent rester secrètes, contrairement aux clés publiques, qui doivent être communiquées.

4.9.2 Serveur

On commence par lancer le démon `sshd`. Démon `sshd` à l'écoute des connexions clientes.

On génère une clé `ssh` pour le compte du serveur, on utilise le programme de génération de clés de `ssh` (`ssh-keygen`). On copie la clé publique fraîchement générée du serveur dans le compte client dans le fichier `.ssh/authorized_keys`,

4.10 Testons la connexion

`ssh` - OpenSSH SSH client (remote login program)

SYNOPSIS

```
ssh [-l login_name] hostname | user@hostname [command]

ssh [-afgknqstvxACNTX1246] [-b bind_address] [-c cipher_spec]
    [-e escape_char] [-i identity_file] [-l login_name] [-m mac_spec]
    [-o option] [-p port] [-F configfile] [-L port:host:hostport]
    [-R port:host:hostport] [-D port] hostname | user@hostname [command]
```

premiers test :

Testons la connexion `ssh` sur le serveur.

Vous êtes le client `vpn` :

```
vpn@b710pbw:~/.ssh$ ssh
```

ou

```
vpn@b710pbw:~/.ssh$ ssh vpn@serveur ls
```

La commande vous demande d'ajouter le site à la liste des machines connues. Vous répondez `yes`.

```
"The authenticity of host '134.214.90.20' can't be established.
Key fingerprint is 1024 f1:ac:4a:48:27:cc:3f:a6:fa:b1:6e:05:c8:35:bf:b7.
Are you sure you want to continue connecting (yes/no)?"
```

Yes

Ensuite, la commande vous demande le mot de passe qui n'est pas la "passphrase" mais simplement le mot de passe unix de votre compte sur le serveur. Une fois la connexion authentifiée, la commande `ls` s'exécute sur le serveur et vous liste les fichiers et répertoires de votre compte sur le serveur.

4.10.1 Copie (utilisation de la commande scp) :

SYNOPSIS

```
scp [-pqrvc46] [-S program] [-P port] [-c cipher] [-i identity_file]
    [-o option] [[user@]host1:]file1 [...] [[user@]host2:]file2
```

exemple : copie de tous les fichiers .tar dans le répertoire /tmp de la machine distante

```
scp *.tar machine@domaine.fr:/tmp
```

4.10.2 Tests avec Ethereal

Test numero 1 : connexion avec telnet : Si on lance sur le réseau un « sniffeur » tel que ethereal, dans la mesure où la transmission des informations s'effectue en clair nous avons pu capturer l'ensemble des données transitant par ce canal, y compris le couple login/mot de passe de l'utilisateur ainsi que les données liées à la commande su à savoir le mot de passe de l'administrateur du serveur . C'est le principal inconvénient de ce service.

Une sage décision est donc de ne pas lancer ce service au démarrage du serveur en commentant la ligne ci-dessous du fichier /etc/inetd.conf

```
#telnet      stream  tcp      nowait  telnetd.telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
```

Test numero 2 : connexion avec ssh : On peut constater que les données échangées entre les deux machines sont que des chiffres et des caractères d'aucune signification pour le hacker!!, donc on ne peut pas intercepter le mot de passe ni les données transmises.

Une autre solution consiste à désinstaller le package *telnetd* en appliquant la commande suivante `dpkg -P telnetd` et à vérifier que le service ne tourne plus sur le serveur.

4.10.3 Déconnexion

Et fermeture de la connexion via la commande `exit` .

```
$ exit
exit
Connection closed by foreign host.
```

4.11 Exemple réel de session SSH

Voici un exemple concret, l'utilisateur `h-etie00` du serveur `bat710.univ-lyon1.fr` désire établir une connexion SSH avec le serveur distant `fcn.homeip.net` sur lequel il possède le compte utilisateur `cyberzoide`.

```
1 #####$ ssh -l cyberzoide fcn.homeip.net
2 cyberzoide@fcn.homeip.net's password:
3 [cyberzoide@FCN ]$ ls
4 developpez/ tmp/
5 [cyberzoide@FCN ]$ cd developpez
6 [cyberzoide@FCN developpez]$ scp *.html h-etie00@bat710.univ-lyon1.fr:/home/h-etie00/public_html/
7 h-etie00@bat710.univ-lyon1.fr's password:
8 all.html          100% |*****| 29856   00:00
9 documentation.html 100% |*****| 19484   00:00
10 faq.css           100% |*****| 7532    00:00
11 fichiers.html     100% |*****| 23651   00:00
12 index.html        100% |*****| 20753   00:00
```

```
13 session.html          100% |*****| 19251    00:00
14 variables.html       100% |*****| 19251    00:00
15 [cyberzoide@FCN developpez]$ ./build
16 [cyberzoide@FCN developpez]$ exit
17 [Connection to fcn.homeip.net closed.
```

L'utilisateur `h-etie00` lance l'utilitaire SSH pour se connecter au serveur `fcn.homeip.net` en tant que `cyberzoide` (1). Le serveur distant demande le mot de passe associé à l'utilisateur `cyberzoide` (2). Une fois identifié, l'utilisateur a recours aux commandes du système distant : `ls` (listage des fichiers) et `cd` (changement de répertoire) (3 à 5). Ensuite, il copie les fichiers `/home/cyberzoide/developpez/*.html` du serveur distant vers le répertoire `/home/h-etie00/public_html/` du compte utilisateur `h-etie00` du serveur `bat710.univ-lyon1.fr` (6). Le serveur `bat710.univ-lyon1.fr` demande le mot de passe associé à l'utilisateur `h-etie00` (7). Puis chacun des fichiers est copié (8 à 14). On lance l'exécution du programme `./build` (15), et enfin on ferme la connexion (16 et 17).

Chapitre 5

Scanning

5.1 Présentation

Nmap est un outil d'exploration de réseau et un scanner de port très utile pour la sécurité

5.1.1 Installation

Copie de *NMAP 3.00* dans le répertoire où l'on souhaite l'utiliser.

```
tar -xvzf nmap-3.00.tgz
cd nmap-3.00
./configure
make
make install
```

5.1.2 Description

Nmap est conçu pour permettre aux administrateurs système ainsi qu'aux personnes curieuses de scanner de larges réseaux pour déterminer quels sont les hôtes connectés et quels services ils offrent.

Nmap supporte un grand nombre de techniques de scan tel que : UDP, TCP connect()¹, TCP SYN (mi-ouvert)², ftp proxy (bounce attack)³, reverse-ident⁴, ICMP (ping sweep)⁵, ACK sweep⁶, FIN⁷

¹Connection TCP standard

²Appelée scan demi-ouvert car il ne s'agit pas d'une vraie connexion, cette technique consiste à envoyer un SYN à un port de la machine cible, et si celle-ci renvoie en réponse un SYN|ACK, c'est que ce port est ouvert (LISTEN). Il faut ensuite envoyer immédiatement un RST afin d'annuler cette connexion.

³Cette technique utilise la "fonctionnalité" proxy d'un serveur FTP (décrite dans le RCF 959) qui permet de se connecter à n'importe quel serveur en passant par ce serveur FTP. Cette méthode permet de masquer l'identité de l'attaquant, les paquets contenant l'adresse du serveur FTP qui a été utilisé comme proxy mais elle est très lente.

⁴Cette méthode utilise le protocole ident (RFC 1413) qui permet de connaître le propriétaire de n'importe quel processus d'une connexion TCP, même s'il ne s'agit pas de l'initiateur. Mais cette méthode requiert une connexion TCP complète ce qui est facilement détectable.

⁵Cette technique est basée sur l'envoi de messages ICMP ECHO request. L'avantage est le fait de reconnaître facilement le réseau ciblé grâce au broadcast ICMP. Mais la plupart des firewalls sont configurés pour bloquer le trafic ICMP ECHO.

⁶On envoie des paquets TCP ACK et on attend les réponses. Un hôte présent doit répondre. Ceci permet de passer à travers des réseaux n'autorisant pas le ping.

⁷L'idée est que les ports fermés doivent répondre par un RST, alors que ceux ouverts ne doivent pas prendre en compte le paquet en question (RFC 793 pp 64).Le scan FIN utilise un paquet avec le flag FIN.

, Xmas tree⁸, SYN sweep⁹, et NULL scan¹⁰. Nmap possède aussi un certain nombre de fonctions avancées telles que la détection de l'OS distant grâce à l'empreinte TCP/IP¹¹, le stealth scanning¹², le calcul de retransmission, le scan en parallèle, la détection d'hôtes éteints grâce au ping en parallèle, les scans avec leurres, la détection de ports filtrés, le scan RPC direct (sans-portmap), le scan avec fragmentations de paquets et une flexibilité dans l'écriture des cibles et des ports.

Le résultat de l'exécution de nmap nous donne normalement une liste de ports intéressants sur la machine scannée.

Nmap donne toujours le nom du service correspondant (s'il est connu), le numéro, l'état, et le protocole. L'état est soit 'open', 'filtered', 'unfiltered'.

- "Open" signifie que la cible acceptera - 'accept()' - des connexions sur ce port.
- "Filtered" signifie qu'il y a un firewall/filtre ou un obstacle qui couvre ce port et qui empêche nmap de déterminer s'il est ouvert ou non. Unfiltered signifie que le port est connu par nmap pour être fermé et qu'aucun firewall/filtre n'apparaît interférer avec nmap.
- Les résultats "unfiltered" sont courants et sont affichés uniquement quand la plupart des ports sont trouvés à l'état filtré.

En fonction des options utilisées, les caractéristiques suivantes peuvent être affichées :

- Le type d'OS.
- Le séquençement TCP.
- Les noms des utilisateurs sous lesquels fonctionnent les programmes ayant ouvert des ports.
- Le nom DNS.
- ...

5.2 Les résultats de NMAP

5.2.1 Première prise en main

Commande : `./nmap-O -v 134.214.90.20`

Résultats :

```
Host (134.214.90.20) appears to be up ... good.
Initiating SYN Stealth Scan against (134.214.90.20)
The SYN Stealth Scan took 169 seconds to scan 1601 ports.
Warning: OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
All 1601 scanned ports on (134.214.90.20) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=3.00%P=i686-pc-linux-gnu%D=10/25%Time=3DB8F5E6%O=-1%C=-1)
T5(Resp=N)
T6(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(Resp=N)
```

⁸Même principe que FIN mais Xmas utilise la combinaison des flags FIN, URG et PUSH.

⁹Cette méthode envoie un SYN plutôt qu'un ACK. L'hôte présent doit répondre par un RST (très rarement par un SYN|ACK).

¹⁰Même principe que FIN mais le scan NULL ne met aucun flag.

¹¹Cette technique exploite les différences entre les implémentations de la pile TCP/IP dans les OS.

¹²Scan furtif qui a les caractéristiques suivantes : paramètre les flags, passe à travers les filtres, firewalls, routeurs, apparaît comme du trafic réseau fortuit.

Nmap run completed -- 1 IP address (1 host up) scanned in 191 seconds

5.2.2 Test sur un routeur autorisant des entrées

Commande : `./nmap-P0 -v 134.214.90.8`

Résultats :

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
No tcp,udp, or ICMP scantype specified, assuming SYN Stealth scan.
Use -sP if you really don't want to portscan (and just want to see
what hosts are up).
WARNING! The following files exist and are readable:
/usr/local/share/nmap/nmap-services and ./nmap-services.
I am choosing /usr/local/share/nmap/nmap-services for security reasons.
  set NMAPDIR=. to give priority to files in your local directory
Host b710pbj.univ-lyon1.fr (134.214.90.8) appears to be up ... good.
Initiating SYN Stealth Scan against b710pbj.univ-lyon1.fr (134.214.90.8)
Adding open port 2003/tcp
Adding open port 23/tcp
Adding open port 37/tcp
Adding open port 21/tcp
Adding open port 512/tcp
Adding open port 25/tcp
Adding open port 514/tcp
Adding open port 13/tcp
Adding open port 513/tcp
Adding open port 113/tcp
Adding open port 755/tcp
Adding open port 6000/tcp
Adding open port 9/tcp
Adding open port 79/tcp
Adding open port 111/tcp
The SYN Stealth Scan took 0 seconds to scan 1601 ports.
For OSScan assuming that port 9 is open and port 1 is closed and neither
are firewalled
Interesting ports on b710pbj.univ-lyon1.fr (134.214.90.8):
(The 1586 ports scanned but not shown below are in state: closed)
Port      State      Service
9/tcp     open      discard
13/tcp    open      daytime
21/tcp    open      ftp
23/tcp    open      telnet
25/tcp    open      smtp
37/tcp    open      time
79/tcp    open      finger
111/tcp   open      sunrpc
113/tcp   open      auth
```

```

512/tcp    open      exec
513/tcp    open      login
514/tcp    open      shell
755/tcp    open      unknown
2003/tcp   open      cfingerd
6000/tcp   open      X11
Remote operating system guess: Linux 2.1.19 - 2.2.20
Uptime 0.085 days (since Fri Oct 25 09:31:40 2002)
TCP Sequence Prediction: Class=random positive increments
                        Difficulty=1358183 (Good luck!)
IPID Sequence Generation: Incremental

```

Explication : On peut voir toute la liste de ports et de services disponibles sur la machine 134.214.90.8.

5.2.3 Cacher l'adresse d'envoi des requêtes

Commande : `./nmap-F -D 134.214.0.1 -v 134.214.90.14`

Petite explication : L'adresse 134.214.0.1 est une adresse "bidon" qui permet de disperser les adresses d'attaques des machines. On peut donc noyer l'adresse source réelle d'attaque dans une nuée de fausses adresses. Cette option permet en effet de mettre le nombre d'adresses bidon que l'on souhaite. De plus, dans le cadre d'un "espionnage" entre entreprises, on pourrait par exemple mettre une adresse d'une autre entreprise concurrente afin de diviser les soupçons.

Résultat : Voici un exemple de ce que etheral a reçu :

No.	Time	Source	Destination	Protocol	Info
1	0.000000	134.214.90.8	134.214.90.14	TCP	Ethernet II
2	0.000061	134.214.0.1	134.214.90.14	TCP	Ethernet II
3	0.000132	134.214.90.8	134.214.90.14	TCP	Ethernet II
4	0.000203	134.214.0.1	134.214.90.14	TCP	Ethernet II
5	0.000275	134.214.90.8	134.214.90.14	TCP	Ethernet II
6	0.000344	134.214.0.1	134.214.90.14	TCP	Ethernet II

Remarque : On constate que les trames ont deux sources possibles : 134.214.90.8 qui est l'adresse de la machine qui a lancé le scan et 134.214.0.1 qui est l'adresse "bidon" qui a été passée en paramètre.

5.2.4 Scanner un réseau

Commande : `./nmap -v 134.214.90.[0-255]`

Résultat :

```

[...]
Host (134.214.90.10) appears to be down, skipping it.
Host (134.214.90.11) appears to be down, skipping it.
Host (134.214.90.12) appears to be down, skipping it.
Host (134.214.90.13) appears to be down, skipping it.
Host (134.214.90.14) appears to be up ... good.
Initiating SYN Stealth Scan against (134.214.90.14)
[...]

```

En fin de scan, il écrit la ligne suivante :

```
Nmap run completed -- 256 IP addresses (8 hosts up) scanned in 455 seconds
```

Explication : On peut de cette manière scanner un réseau automatiquement. Nmap va tester chaque machine et afficher les descriptions des ports ouverts quand les machines répondent. Pour les machines qui ne répondent pas, il va les signaler comme dans l'exemple ci-dessus.

Chapitre 6

Rootkits et Trojans

6.1 Définitions

Rootkit Ensemble d'exploits réunis afin d'avoir des chances maximales de prendre le contrôle d'un compte root sur une machine Unix. Un rootkit est donc une collection de fichiers/programmes employée par l'attaquant pour pénétrer un réseau/ordinateur sans être détecté.

Trojans Ces petits programmes vont faire des actions très spécifiques sur une machine. Ils sont utilisés par les rootkits pour modifier des fichiers de commande tels que `ls`, `netstat`, `top` afin de cacher un hôte indésirable. Une autre application très répandue des trojans est par exemple d'infecter le fichier `/bin/login` et de donner à certains utilisateurs les privilèges de root.

Backdoor Porte d'entrée cachée dans un système. C'est souvent une porte de service discrète mais délibérément laissée par ses concepteurs ou ses réalisateurs, qui servait lors du débogage et a été oubliée. Ce type de porte peut aussi être implanté par un hacker pour lui permettre de revenir dans un système sans avoir à le repirater.

Exploit Description d'une technique de piratage, d'un « truc » permettant d'exploiter une faille de sécurité.

Script kiddy Hacker "juste" capable (sans comprendre ce qui se passe réellement) de faire tourner une poignée de scripts pouvant lui donner un accès non autorisé à un système.

6.2 Résumé : étapes du hacker

1. Recherche d'une proie : scan de machines. Le hacker utilise alors un outil de scan.
2. Pratique de l'exploit : le hacker prend le contrôle de la machine en tant que root à l'aide d'un rootkit qui exploite une faille de sécurité connue du système. On aura alors besoin d'un rootkit connaissant plusieurs exploits.
3. Se cacher : installation de trojans et effacement des fichiers de log. Cette étape nécessite les trojans du rootkit et des outils de nettoyage.
4. Préparer ses prochaines visites : divers trojans (actifs ou passifs) installent des backdoors pour pouvoir se reconnecter en tant que root sans problèmes. Pour ce faire, le hacker utilise les autres outils et fonctionnalités du rootkit.
5. 3 possibilités : Premièrement, le hacker utilise le système comme plate-forme de lancement d'attaque en scannant ou en forçant d'autres systèmes. Autrement, il peut tenter d'étendre sa prise en cherchant des informations pour connaître les mots de passe des comptes utilisateurs. La dernière

possibilité est de commettre des actions destructrices (effacement de fichiers, vol de données,...) sur la machine à laquelle il a déjà accès, au risque de ne plus pouvoir se réintroduire dans le système. Cette étape peut nécessiter des sniffers ou des outils de scan selon les choix d'actions à mener.

6.3 Mon expérimentation

Le but d'un rootkit est de pénétrer un système et de s'y installer. Le rootkit va essayer de placer un programme sur une machine cible qui laisse une porte ouverte (backdoor) afin de (re)prendre son contrôle à distance en temps voulu.

J'ai essayé une bonne douzaine de programmes différents récupérés sur le net par l'intermédiaire de sites plus ou moins douteux. Mais aucun n'a fonctionné du début à la fin. C'est à dire que je n'ai pas réussi à prendre le contrôle d'une machine à distance. Les différents problèmes que j'ai rencontrés étaient des problèmes de documentation des programmes et des problèmes de bibliothèques non disponibles sur les machines dédiées aux TPs.

Mais j'ai tout de même pu observer le principe de base du rootkit et une partie des actions que ces programmes peuvent faire. Ceci laisse donc supposer que ce risque d'attaque est tout à fait possible.

Bien sûr, dans le monde réel (je veux dire le monde où je ne suis pas root), une des majeures difficultés pour l'assaillant est de pénétrer le système pour installer une backdoor sur la machine cible. Dans mon cas je n'ai pas exploité de faille de sécurité des systèmes. J'ai juste tenté d'installer la partie du programme qui doit donner une porte d'entrée facile au hacker et cacher le programme à la vue de l'administrateur.

Il est à noter que les attaquants potentiels ne vont pas forcément se tourner vers une cible très intéressante tel qu'un serveur mais plutôt vers un client anonyme pour ensuite attaquer le serveur après infiltration. Ceci signifie donc que toutes les machines doivent être bien protégées car la sécurité globale d'un système d'information dépend de son maillon le plus faible.

6.4 Comment se présente le programme ?

La plupart du temps, un rootkit contiendra divers exploits pour aider l'attaquant à pénétrer un système.

En plus des exploits, beaucoup de rootkits également contiennent et installent des sniffers. Ceci est fait parce que les attaquants veulent le plus souvent capturer les mots de passe des utilisateurs entrant sur ce réseau afin d'étendre leur prise sur le système et d'accéder aux données convoitées.

Le rootkit contient aussi divers trojans qui cachent leur existence sur le système infecté et qui installent des backdoors.

6.5 Mode de fonctionnement

Tout d'abord, en ce qui concerne la pratique de l'exploit, il est différent selon chaque système. Les exploits utilisent les failles de sécurité des navigateurs, des systèmes d'exploitation, des serveurs http....

Le rootkit va ensuite installer son code sur la machine et mettre en route ses processus. Il doit alors se protéger. Le principal mode de protection est de se cacher. Ce mode de protection est très efficace et la plupart des rootkits sont très très difficiles à détecter. En effet, bien qu'il existe des programmes qui permettent de scanner la machine et ces différentes fonctionnalités pour voir si d'éventuels rootkits ou trojans sont installés, le résultat n'est pas toujours celui escompté.

Il existe deux parties du rootkit à cacher : le code physique et les processus en cours de fonctionnement. Comme que la plupart des trojans sont actifs (ils attendent d'être utilisés en tant que backdoor), ils doivent donc se cacher dans la table des processus en activité. Pour cette raison, certaines routines sont chargées de détruire tout simplement la commande ps ou d'autres, plus discrètement la modifient. De même, le code du programme doit être caché et la commande ls peut alors elle aussi être modifiée.

Les rootkits ont donc l'habitude de changer les binaires communs de sorte qu'un administrateur occupé ne les détecte pas. Les binaires les plus souvent hackés sont les binaires qui peuvent être employés pour surveiller les systèmes : /bin/ps, /bin/ls, /bin/netstat, /usr/bin/lsof, /usr/bin/top (ce n'est pas une liste complète).

Pour ce qui est de la manière dont le rootkit installe les backdoors, il s'agit le plus souvent d'un daemon qui écoute sur un port spécifique. Il suffit alors de faire un telnet sur ce port avec un login connu de celui qui a installé le rootkit pour prendre le contrôle de la machine. On pourra ensuite visiter les comptes utilisateurs et tenter de récupérer leurs mots de passe, installer un sniffer et analyser le trafic pour découvrir ces mêmes mots de passe ou attaquer directement un serveur si on le souhaite.

6.6 Top du moment

Un des rootkit le plus utilisé est lkm4 (version 4). J'ai essayé de l'utiliser mais il n'a pas pu s'installer à cause d'un manque de bibliothèque. LKM signifie modules du noyau chargeables (Loadable Kernel Modules).

Dans le monde de Linux, LKM sont des morceaux de code qui peuvent être chargés dans le système d'exploitation sur demande. LKM est employé pour mettre à jour le matériel pour l'OS.

Le fait qu'LKM utilise ce principe pour se cacher le rend quasiment indécélable. Ce nouveau système est beaucoup plus performant. Les programmes sensés vérifier si les machines sont victimes de rootkit ne sont plus désormais capables de déceler les programmes cachés grâce à ce système. En effet, cela ne suffit pas de vérifier chacune des fonctionnalités. Les programmes doivent revoir leur stratégie de détection.

6.7 Les protections existantes

Il est à noter qu'il n'existe pas de réelle protection infaillible. En effet, les failles de sécurité ne seront jamais toutes découvertes. Seule une surveillance de la part de l'administrateur peut permettre de limiter les intrusions. Voici quelques règles de base qui peuvent permettre de "limiter la casse" :

- Appliquer les mises à jours des logiciels (patches qui corrigent les failles).
- Utiliser régulièrement des outils de contrôles (Scan des ports ouverts et tests des commandes systèmes).
- Tester sa sécurité (Nmap, Nessus)

Chapitre 7

Audit de réseau avec nessus

7.1 Introduction

Nessus est un outil de test de vulnérabilité. Il fonctionne en mode client/serveur, avec une interface graphique.

7.2 Sources

Les sources et la documentation sont disponibles à l'adresse <http://www.nessus.org>

Pour Gnu/Linux, il faut les fichiers

- libnasl-1.2.6.tar.gz et
- nessus-libraries-1.2.6.tar.gz, ce sont 2 bibliothèques,
- nessus-core-1.2.6.tar.gz, les fichiers de nessus proprement dit,
- nessus-plugins-1.2.6.tar.gz, les attaques utilisées par nessus.

Openssl est nécessaire si on veut utiliser une authentification du client par le serveur en utilisant le chiffrement.

7.3 Installation

Installer dans l'ordre : nessus-libraries, libnasl, nessus-core et nessus-plugins.

On obtient alors

- les binaires
 - /usr/local/bin/nessus
 - /usr/local/bin/nessus-build
 - /usr/local/bin/nessus-config
 - /usr/local/bin/nessus-mkcert-client
 - /usr/local/bin/nessus-mkrand
 - /usr/local/sbin/nessus-adduser
 - /usr/local/sbin/nessus-mkcert
 - /usr/local/sbin/nessus-rmuser
 - /usr/local/sbin/nessus-update-plugins
 - /usr/local/sbin/nessusd
 - /usr/local/sbin/uninstall-nessus
- la documentation dans

- /usr/local/man/man1/
- /usr/local/man/man8/
- les plugins (les attaques)
 - /usr/local/lib/nessus/plugins/
- les enregistrements des résultats dans
 - /usr/local/var/nessus/users/jeff/sessions/

7.4 Configuration

Il faut créer un certificat pour le démon `nessusd`, avec `nessus-mkcert`, et ajouter un utilisateur avec `nessus-adduser` (on peut aussi utiliser un simple mot de passe, sans passer par `mkcert`, auquel cas `openssl` n'est pas nécessaire).

Il existe plusieurs options pour l'utilisateur (règles d'accès).

7.5 Utilisation

L'utilisation de `nessus` est très intuitive et conviviale, grâce à une interface graphique simple et logique.

Lancer d'abord le démon : `nessusd`, éventuellement sur une machine distante. `nessusd` est le serveur, qui va effectuer le test et envoyer les résultats au client.

Lancer alors le client : `nessus`. On obtient une fenêtre avec plusieurs onglets :

- l'onglet de connexion (fig. 7.1) permet de préciser l'adresse de la machine où tourne le démon et le port sur lequel il écoute (par défaut `localhost`, port 1241), ainsi que le nom de l'utilisateur et son mot de passe.
- l'onglet plugins (fig. 7.2) permet de sélectionner les tests que l'on souhaite réaliser.
- l'onglet prefs (fig. 7.3) permet le réglage de différents paramètres pour les tests.
- l'onglet scan-options (fig. 7.4) permet le paramétrage du scan, avec notamment la possibilité de faire des tests inoffensifs (`safe checks`) pour préserver la machine cible.
- l'onglet target selection (fig. 7.5) permet de désigner la (ou les) cible(s) choisie(s), et affiche un historique des adresses précédemment testées.
- l'onglet user (fig. 7.6) permet de définir des règles pour l'utilisateur.
- l'onglet kb (fig. 7.7) offre différentes options d'utilisation de la base de données des tests déjà effectués (pour éviter de répéter des tests par exemple).
- le dernier onglet (fig. 7.8) nous apprend le nom de l'auteur, etc.

Quand on lance le test, cette première fenêtre est remplacée par une autre qui montre l'avancement du test (fig. 7.9).

A la fin du test, apparaît une autre fenêtre (fig. 7.10) qui présente les résultats. Cette présentation est faite de façon très didactique, avec explication des failles découvertes, de leur utilisation possible, de leur gravité et de la façon de les résoudre.

On peut alors enregistrer les résultats sous diverses formes : texte, latex, html, html avec graphiques...

7.6 Tests effectués et résultats observés

Divers essais ont été effectués avec `nessus`, sur notre routeur, à différents stades de la mise en place du pare-feu, et aussi sur d'autres machines.

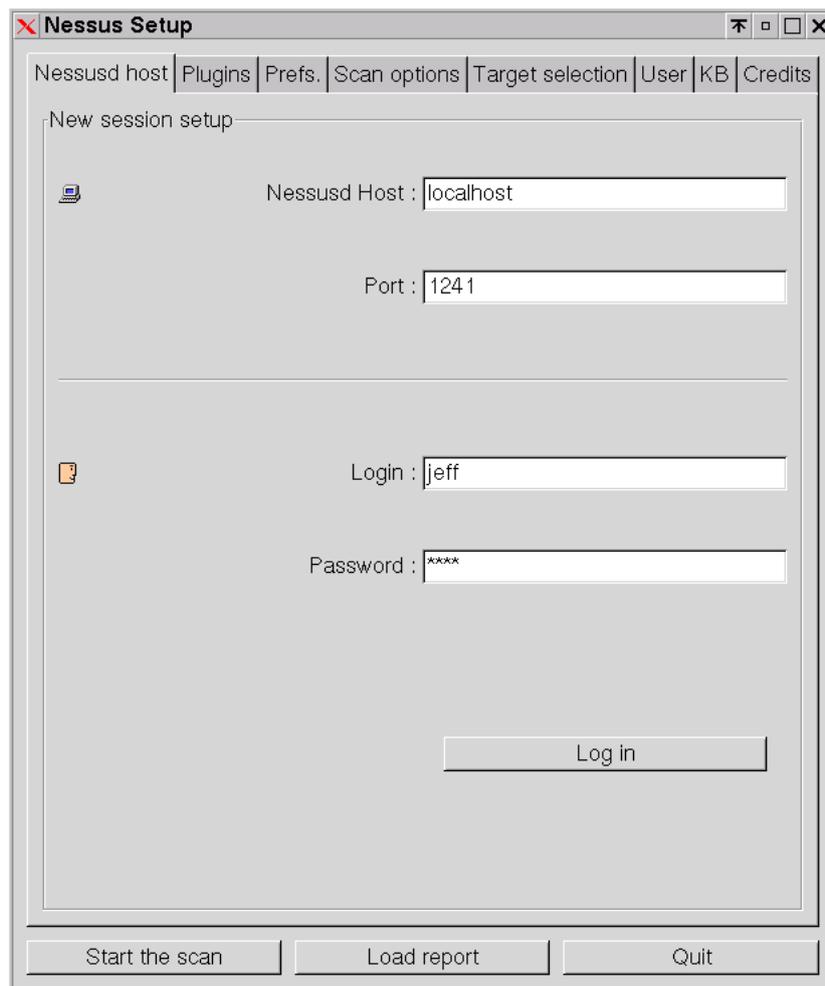


FIG. 7.1 – Onglet de connexion

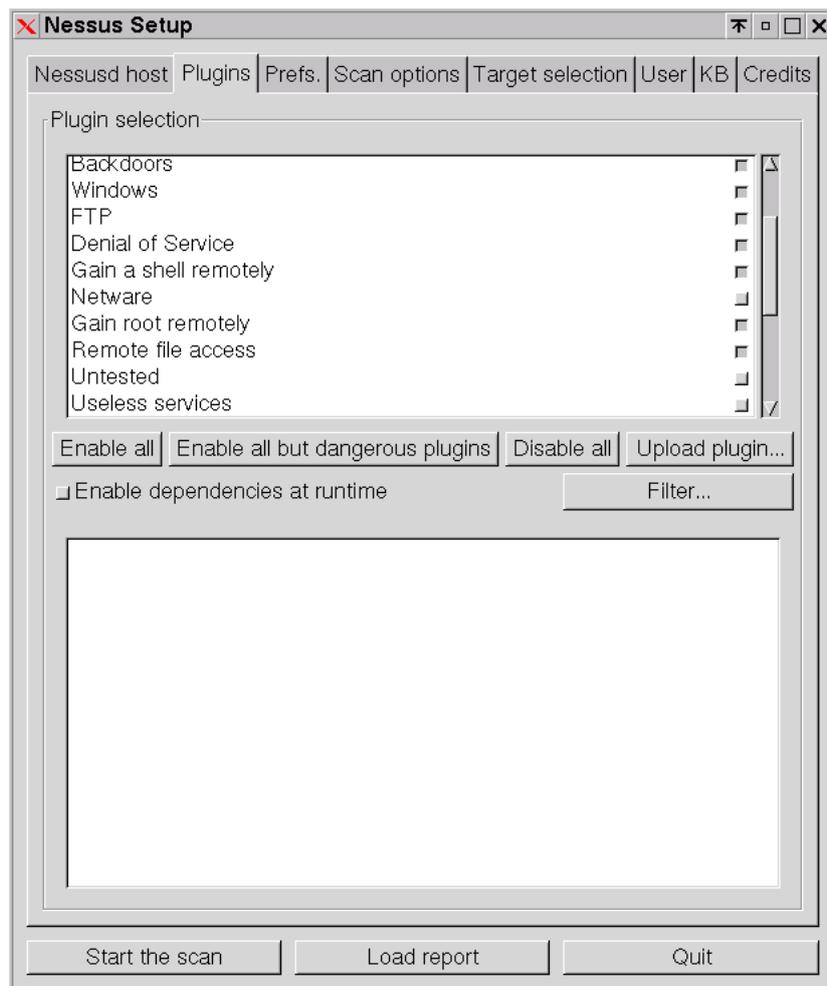


FIG. 7.2 – Onglet plugins

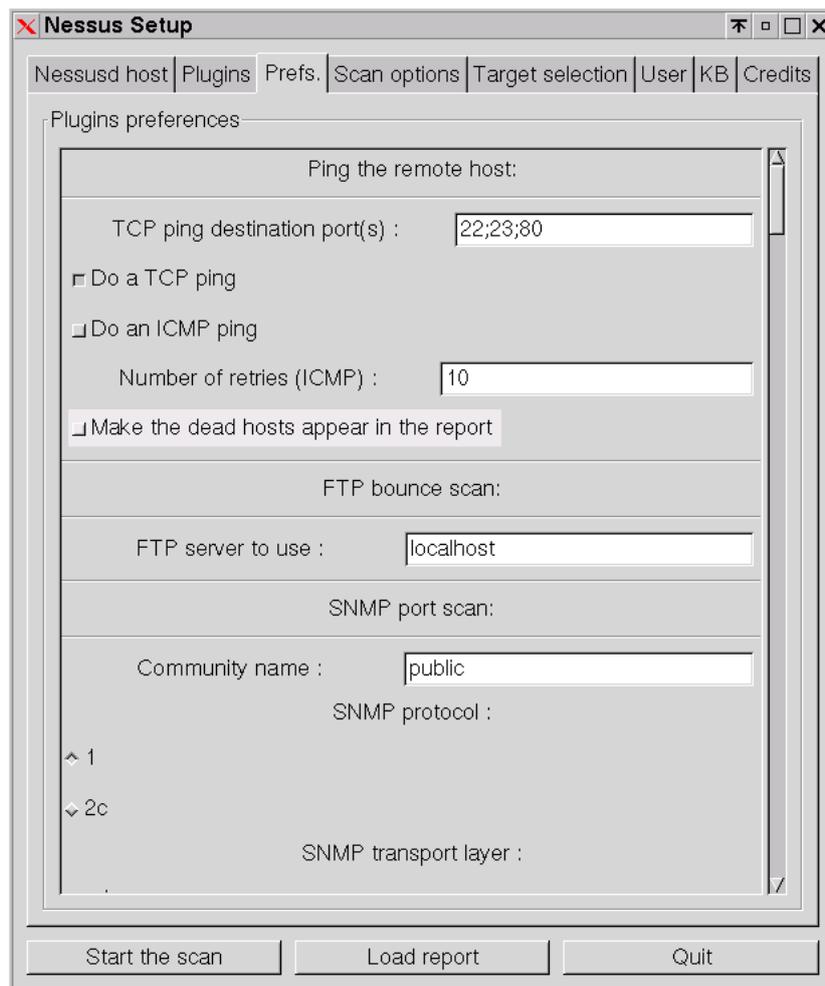


FIG. 7.3 – Onglet de préférences

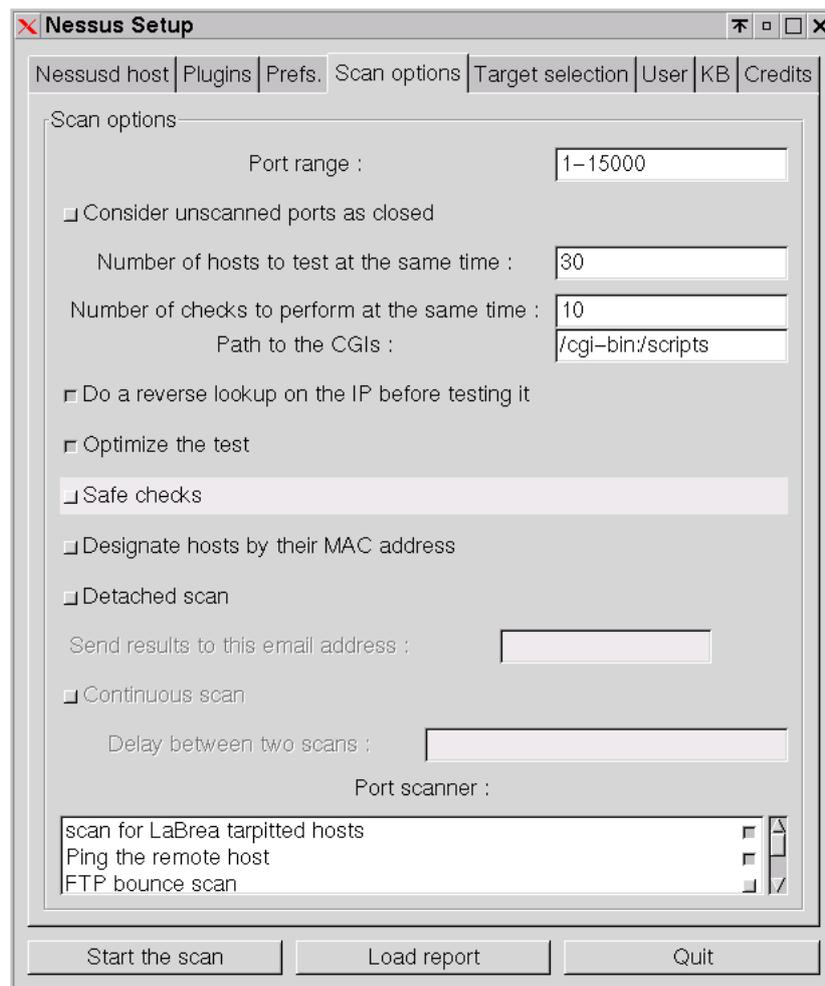


FIG. 7.4 – Onglet des options de scan

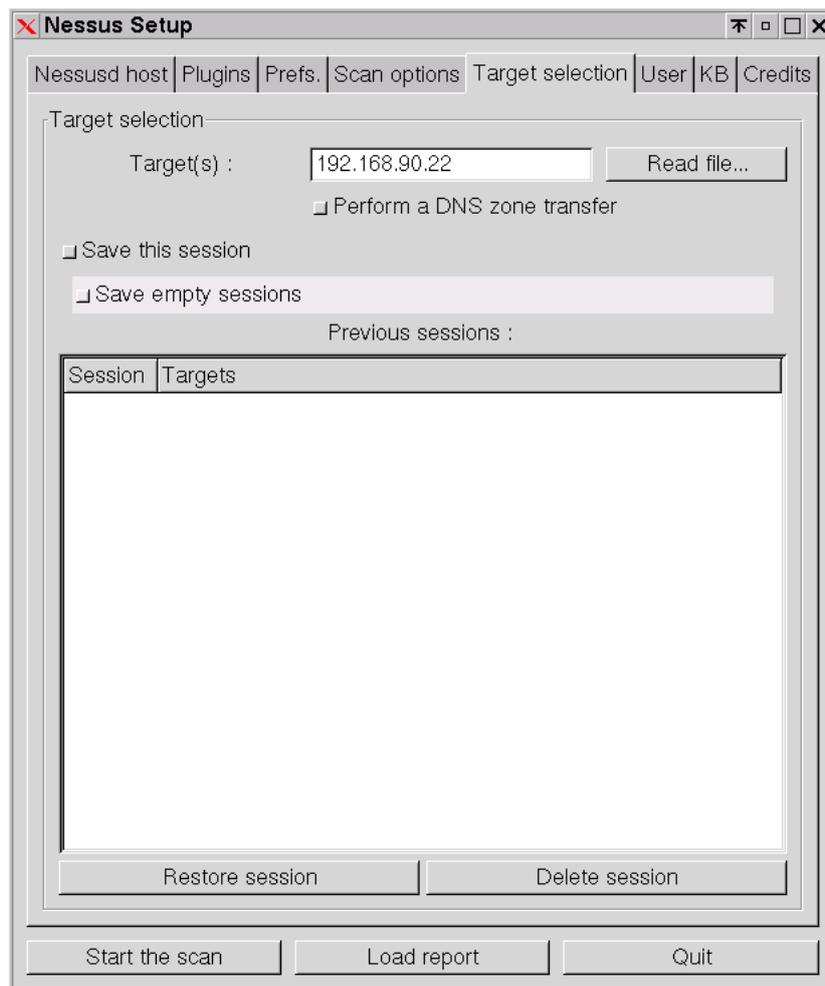


FIG. 7.5 – Onglet de sélection de la cible

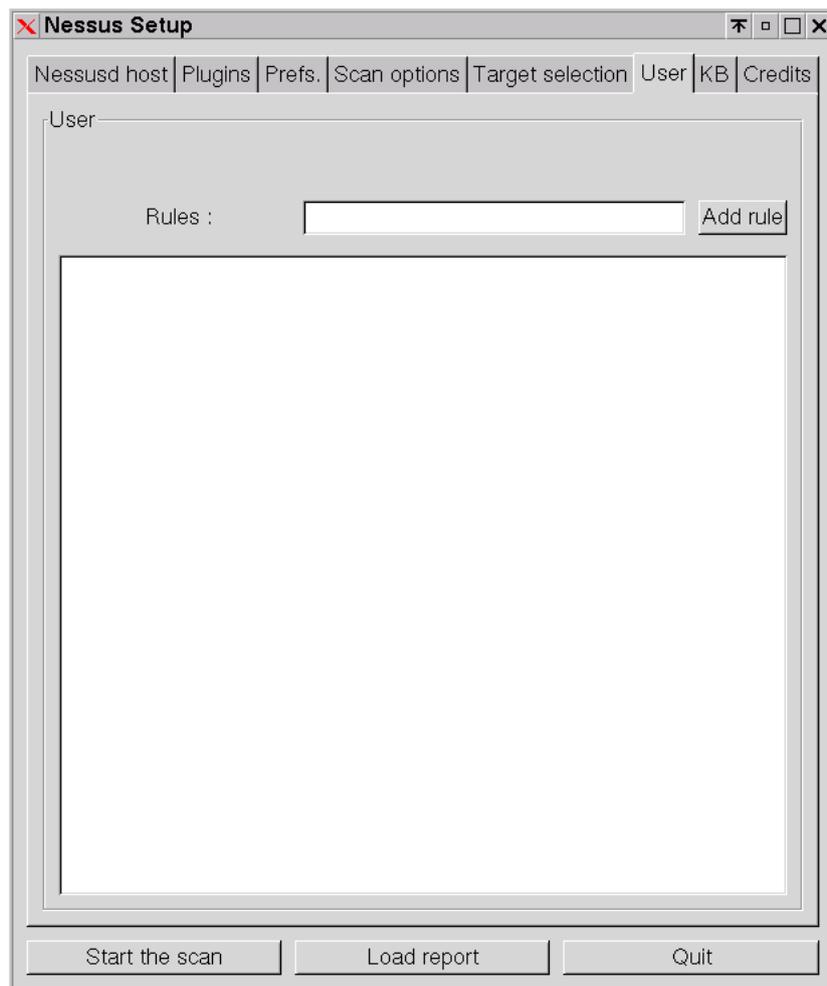


FIG. 7.6 – Onglet de l'utilisateur

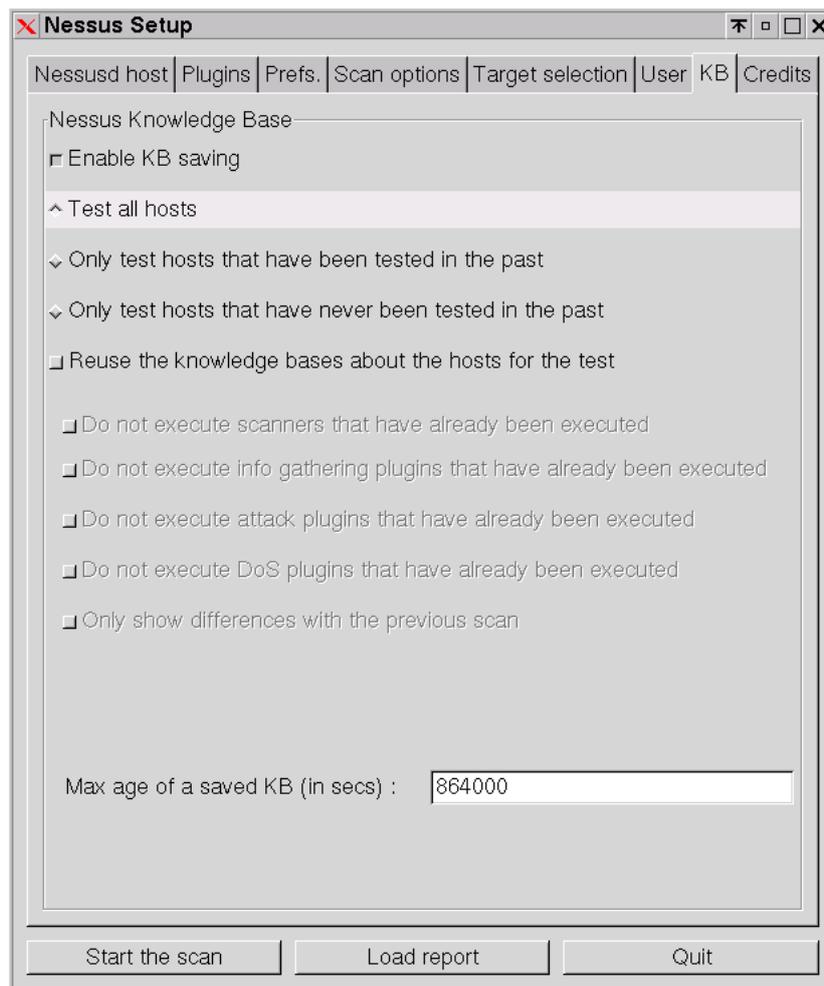


FIG. 7.7 – Onglet de la base de connaissances

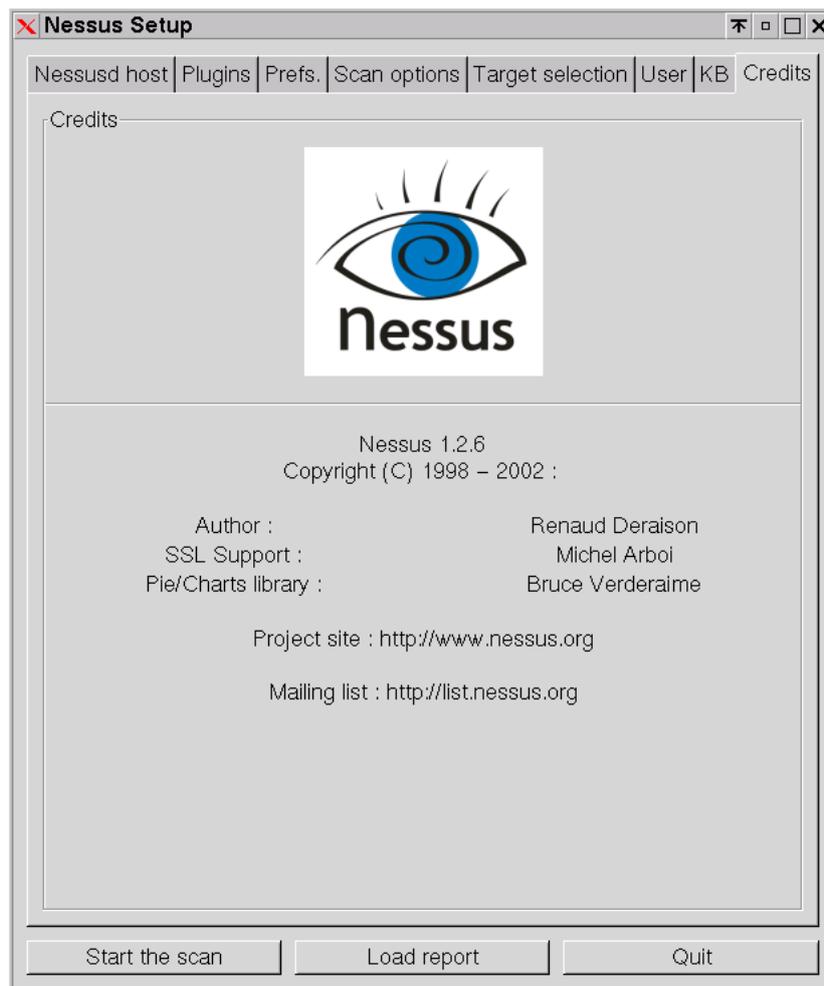


FIG. 7.8 – Onglet crédits

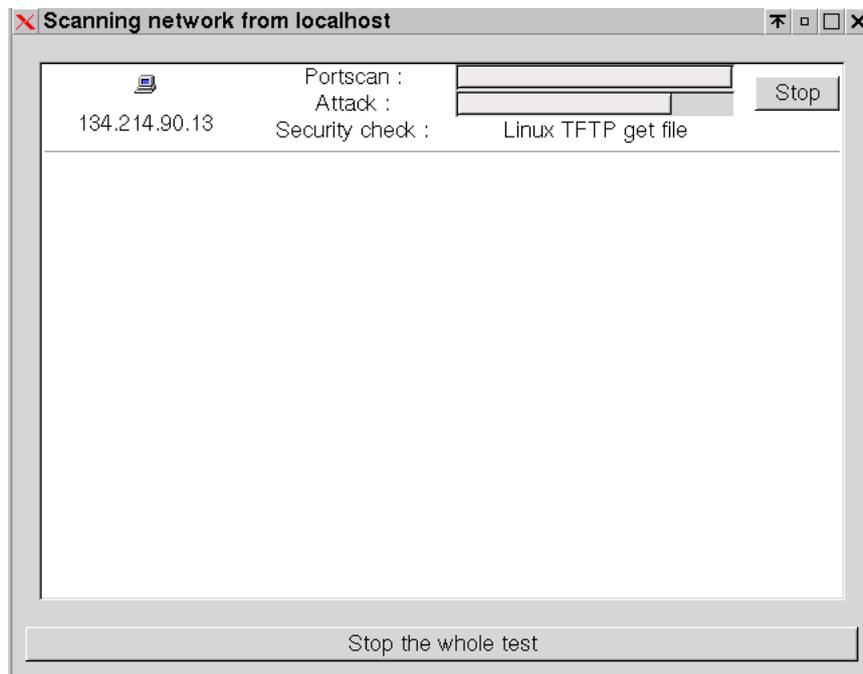


FIG. 7.9 – Fenêtre d'avancement du scan

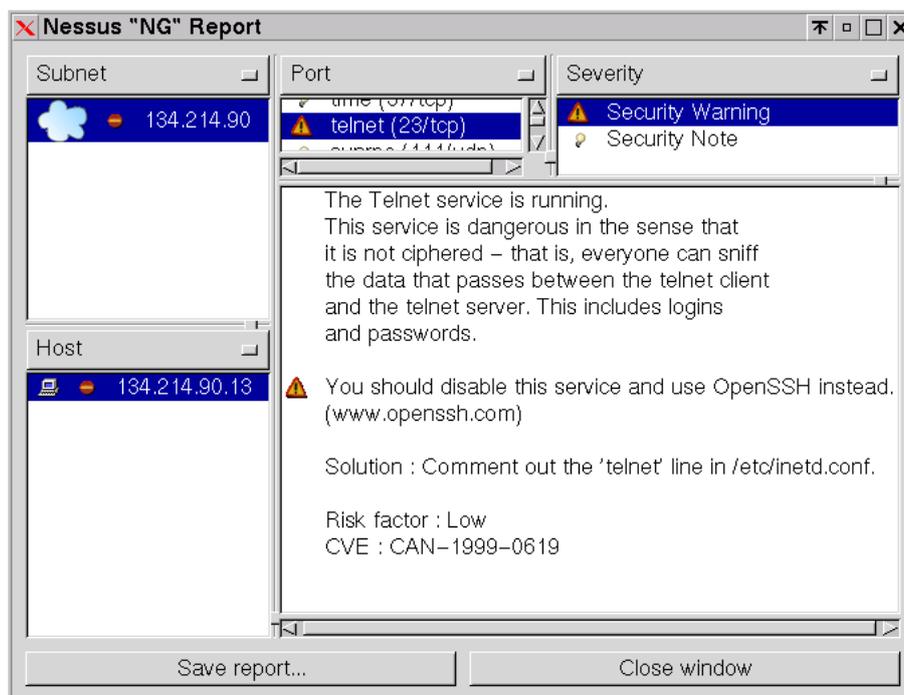


FIG. 7.10 – Résultats de l'audit

7.6.1 Test sur une machine protégée totalement

Sur notre routeur, au moment où il a été protégé par ipchains, et configuré pour ne rien laisser passer, la réaction de nessus est de considérer la machine comme “morte”, car elle ne répond à aucun scan :

(fichier de log)

```
timestamps||scan_start|Fri Nov  8 10:23:32 2002|
timestamps||134.214.90.20|host_start|Fri Nov  8 10:23:33 2002|
results|134.214.90|134.214.90.20|general/tcp|10180|Security Note|The remote
host is considered as dead - not scanning
timestamps||134.214.90.20|host_end|Fri Nov  8 10:26:06 2002|
timestamps||scan_end|Fri Nov  8 10:26:06 2002|
```

7.6.2 Test sur une machine laissant peu de ports accessibles

Quand des ports sont ouverts par le pare-feu au cas par cas, nessus voit qu'ils sont ouverts, et fait des mises en garde contre des problèmes pouvant se poser sur ces ports :

(extraits de fichier de log)

```
SERVER <|> TIME <|> HOST_START <|> 134.214.90.20 <|> Tue Oct 29 11:09:59 2002
<|> SERVER
s:a:134.214.90.20:1:1100
.
.
.
s:a:134.214.90.20:354:1100
SERVER <|> INFO <|> 134.214.90.20 <|> general/icmp <|> \nThe remote host answers to an ICMP
timestamp\nrequest. This allows an attacker to know the\ndate which is set on your machine.
\n\nThis may help him to defeat all your \ntime based authentication protocols.\n\nSolutio
n : filter out the ICMP timestamp\nrequests (13), and the outgoing ICMP \ntimestamp replies
(14).\n\nRisk factor : Low\nCVE : CAN-1999-0524\n <|> 10114 <|> SERVER
s:a:134.214.90.20:355:1100
s:a:134.214.90.20:356:1100
.
.
.
s:a:134.214.90.20:718:1100
s:a:134.214.90.20:719:1100
SERVER <|> NOTE <|> 134.214.90.20 <|> general/udp <|> For your information, here is the tra
ceroute to 134.214.90.20 : \n134.214.90.20\n <|> 10287 <|> SERVER
s:a:134.214.90.20:720:1100
s:a:134.214.90.20:721:1100
.
.
.
SERVER <|> FINISHED <|> 134.214.90.20 <|> SERVER
SERVER <|> TIME <|> HOST_END <|> 134.214.90.20 <|> Tue Oct 29 11:11:35 2002
<|> SERVER
```

Netscape: Nessus Scan Report		
File Edit View Go Communicator		
134.214.90.22	unknown (785/tcp)	Security warning(s) found
134.214.90.22	sunrpc (111/tcp)	Security notes found
134.214.90.22	sunrpc (111/tcp)	Security notes found
134.214.90.22	login (513/tcp)	Security warning(s) found
134.214.90.22	unknown (787/tcp)	Security notes found
134.214.90.22	telnet (23/tcp)	Security warning(s) found
134.214.90.22	general/tcp	Security notes found

Security Issues and Fixes: 134.214.90.22		
Type	Port	Issue and Fix
Vulnerability	ftp (21/tcp)	<p>You are running a version of wu-ftpd which is older or as old as version 2.6.0. These versions do not sanitize the user input properly and allow an intruder to execute arbitrary code through the command SITE EXEC.</p> <p>*** Nessus did not log into this server *** so it could not determine whether the option SITE EXEC was activated or not, so this message may be a false positive</p> <p>Solution : upgrade to wu-ftpd 2.6.1 Risk factor : High CVE : CVE-2000-0573</p>
Vulnerability	ftp (21/tcp)	<p>You seem to be running an FTP server which is vulnerable to the 'glob heap corruption' flaw. An attacker may use this problem to execute arbitrary commands on this host.</p> <p>*** As Nessus solely relied on the banner of the server to issue this warning, *** so this alert might be a false positive</p> <p>Solution : Upgrade your ftp server software to the latest version. Risk factor : High CVE : CVE-2001-0550</p>
Vulnerability	ftp (21/tcp)	<p>It was possible to disable the remote FTP server by connecting to it about 3000 times, with one connection at a time.</p> <p>An attacker may use this flaw to prevent this service from working properly.</p>

FIG. 7.11 – Extrait de l’audit du serveur web enregistré au format HTML

7.6.3 Test sur un serveur web

Un audit effectué à partir du réseau local, sur notre serveur web (apache), a révélé 13 failles (security holes) potentielles concernant les ports des protocoles ftp, smtp et www, ainsi que 15 mises en garde (warnings) sur divers ports.

7.7 Conclusion

Nessus est un outil d’audit très complet, dont nous sommes loin d’avoir évalué toutes les possibilités. Son intérêt est aussi didactique : pour chaque faille, on a une présentation claire du problème et une solution simple.

Cet outil peut très certainement servir à une personne compétente à évaluer les faiblesses d’un réseau en vue d’une attaque, en indiquant quelles failles exploiter et avec quelles techniques.

Mais bien sûr, tout bon administrateur devrait se servir d’un tel outil pour éviter au moins les attaques connues de nessus, qui doivent tout de même être devenues des classiques, pour avoir été inventoriées et implémentées sous forme des plugins ...

Sa principale limite serait donc l’“obsolescence” relative des attaques connues au moment de l’audit, ce qui semble inévitable.

Chapitre 8

Forger des paquets avec μ ito

8.1 Introduction

μ ito (à prononcer muto) est un programme permettant de forger des paquets TCP/IP sous Linux. Les protocoles actuellement pris en charge sont : IP, ICMP, IGMP, TCP et UDP ; en attendant ETH, ARP/RARP, DNS, OSPF et RIP prévus par l'auteur dans le développement futur.

Il s'agit là encore d'un logiciel libre avec une interface graphique intuitive et esthétique.

8.2 Installation

Les sources sont disponibles ici : <http://projet7.tuxfamily.org/factory/releases/uito.tar>, et la documentation là : <http://projet7.tuxfamily.org/factory/articles/uito.html>.

Les sources se présentent en archive tar : μ ito.tar, qui contient les fichiers :

- μ ito.c, le programme,
- clap.xpm, l'image affichée dans la fenêtre du programme.

La ligne de commande pour l'installation est indiquée en tête du fichier c :

```
cc  $\mu$ ito.c -o  $\mu$ ito `gtk-config --cflags --libs`
```

On obtient alors le fichier binaire μ ito.

8.3 Utilisation et test

8.3.1 Utilisation

A l'exécution, μ ito ouvre une fenêtre avec le seul onglet IP (fig. 8.1), dans lequel il faut remplir les champs du paquet avec les valeurs désirées.

Dès lors qu'on a choisi un protocole, parmi la liste disponible (ICMP, IGMP, TCP et UDP), apparaissent deux onglets supplémentaires :

- un onglet propre au protocole (fig. 8.2 et 8.3), avec à nouveau les champs du datagramme à renseigner,
- un onglet 'data' pour la partie 'données' du paquet (fig. 8.4).

De manière générale, tous les champs doivent être remplis, sauf le champ data qui peut être vide. Pour certains champs, on peut utiliser les valeurs 'auto' ou 'random'. Les valeurs attendues doivent être de type décimal, sauf le champs 'données', qui est de type caractère.

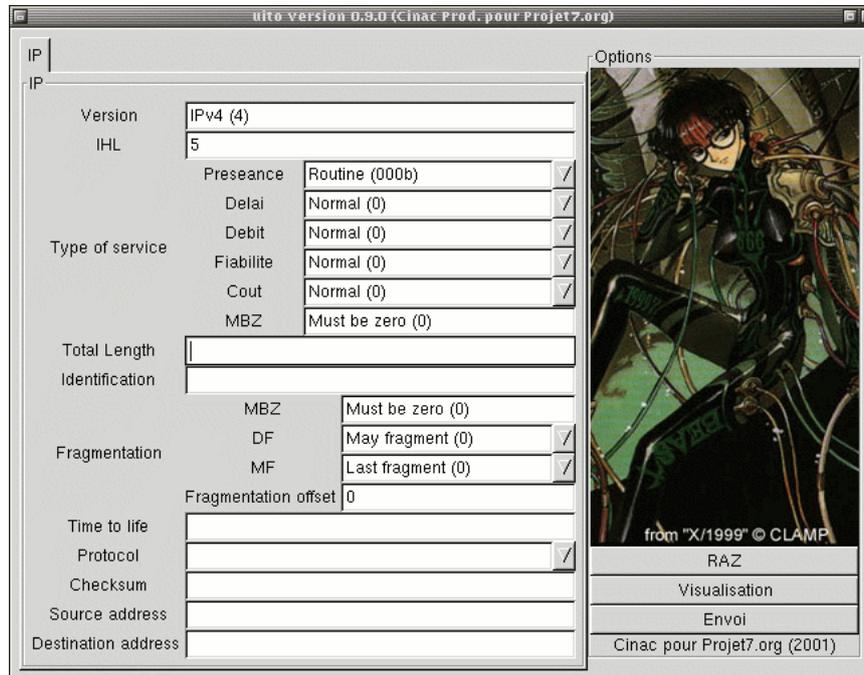


FIG. 8.1 – Onglet ip

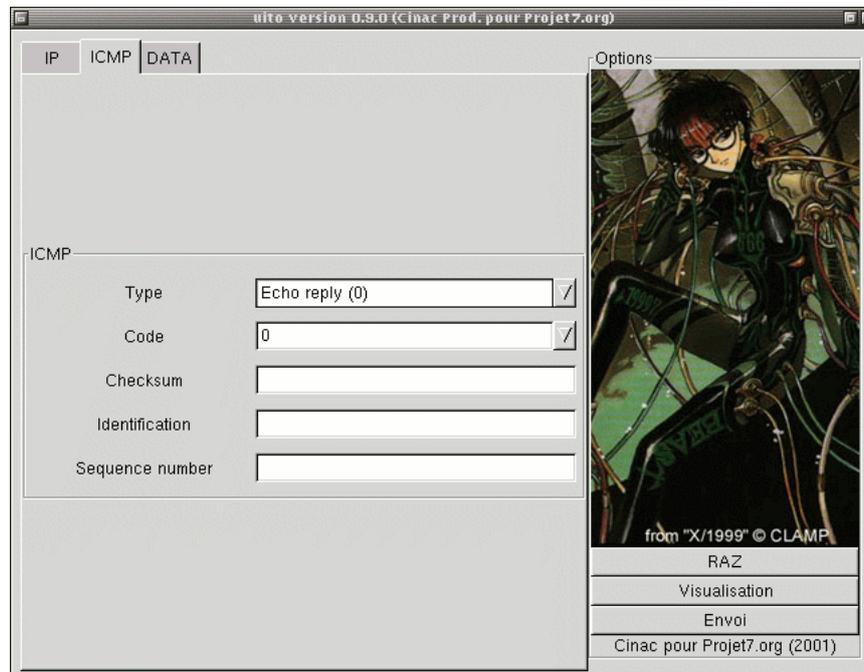


FIG. 8.2 – Onglet icmp

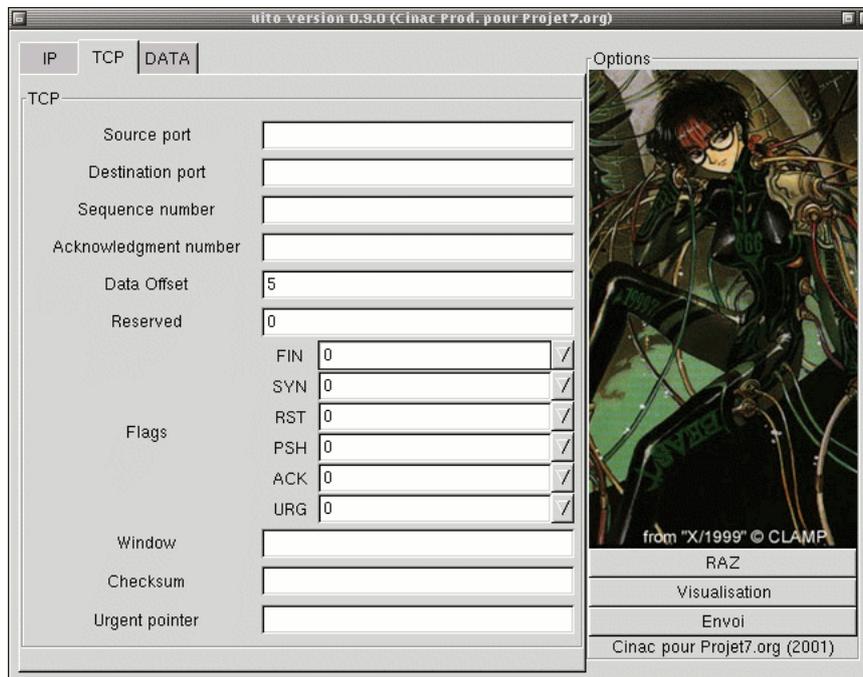


FIG. 8.3 – Onglet tcp

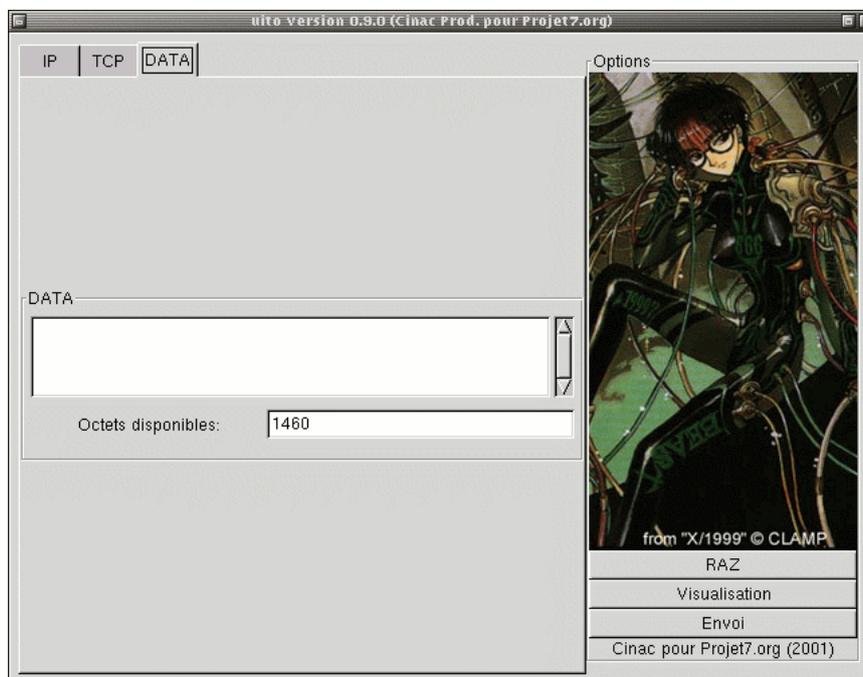


FIG. 8.4 – Onglet données

8.3.2 Test

Nous avons testé μ ito en envoyant un paquet ICMP forgé avec une adresse IP différente de l'adresse réelle : la machine réceptrice croyait effectivement recevoir un ping de la fausse adresse et y répondait.

8.4 Conclusion

μ ito est un très petit logiciel (en poids sur le disque), mais qui offre la possibilité de forger très facilement des paquets.

La principale limitation est qu'on ne peut forger les paquets qu'un à un, rendant difficile l'utilisation dans le cadre d'échanges "falsifiés" (IP spoofing), à cause du temps nécessaire pour chaque paquet.

Chapitre 9

Intranet : Serveur HTTP Apache

A quoi va servir le serveur HTTP Apache? Nous allons utiliser Apache pour mettre en oeuvre un Intranet. Un ou plusieurs sites internes pourront ainsi se développer pour, par exemple, la mise en commun de connaissances ou encore des informations concernant des actions syndicales... Nous justifions la mise en place d'un serveur HTTP par le fait que c'est un élément important d'un réseau pour communiquer et informer (humainement parlant). En effet, quelle entreprise d'une importance moyenne n'a pas un serveur HTTP? Il est donc important de connaître les actions défensives à opérer afin de sécuriser un serveur Apache.

9.1 Choix du serveur

Pourquoi avoir choisi le serveur Apache?

Il s'agit d'un serveur très populaire, performant, et sa conception modulaire le dote d'une grande richesse fonctionnelle (même si nous envisageons de n'utiliser que le module PHP).

De plus, et là encore ce n'est pas rien, le serveur Apache est libre!

Sources : <http://www.apache.org/>

9.2 Installation

L'installation du serveur Apache 1.3.12 est très simple. Il suffit de suivre les instructions du fichier INSTALL.

Voici les commandes d'installation (les privilèges de root sont requis)

NOTE: PREFIX n'est pas la chaîne "PREFIX". Utilisez à la place le chemin sous lequel Apache devrait être installé. Par exemple utilisez "/usr/local/apache" au lieu de PREFIX.

C'est le chemin que nous avons choisi pour notre installation : /usr/local/apache sur la machine 192.168.2.1

```
> ./configure --prefix=/usr/local/apache
> make
> make install
```

```
-----
|You now have successfully built and installed the
| Apache 1.3 HTTP server.To verify that Apache actually
```

```
| works correctly you now should first check the
| (initially created or preserved) configuration files
|
| /usr/local/apache/conf/httpd.conf
|
| and then you should be able to immediately fire up
| Apache the first time by running:
|
| /usr/local/apache/bin/apachectl start
|
| Thanks for using Apache.The Apache Group
| http://www.apache.org/
```

Cette dernière commande effectue l'installation du serveur et nous indique que l'installation s'est bien passée. On nous indique également l'emplacement du fichier de configuration : `/usr/local/apache/conf/httpd.conf`. La manière de démarrer le serveur nous est donnée : `/usr/local/apache/bin/apachectl start`. Faisons-le :

```
> /usr/local/apache/bin/apachectl start
```

Le terminal nous répond : `/usr/local/apache/bin/apachectl start : httpd started`. Apache vient d'être démarré.

9.3 Configuration du serveur

Comme nous l'avons dit, la configuration du serveur Apache se réalise dans le fichier `httpd.conf` seulement (les fichiers `srml.conf` et `access.conf` ne servent plus à rien). Une explication concise du fonctionnement de ce fichier peut se révéler utile. Ce fichier contient des directives relatives au comportement du serveur. Les sections suivantes vont montrer comment utiliser ces directives pour sécuriser le serveur.

9.3.1 Cacher les bannières

Tout d'abord, le premier réflexe est de faire disparaître toute information qui pourrait être nuisible comme la version du serveur et les modules utilisés. Ainsi on cache aux pirates des informations qui auraient pu leur servir à lancer des attaques spécifiques reprenant des failles bien connues. Ces bannières apparaissent à plusieurs endroits selon les distributions Linux. La requête d'une page inexistante renvoie par défaut une page d'erreur 404 avec en bas le message `Apache-AdvancedExtranetServer/1.3.20 Server at 127.0.0.1 Port 80` qui révèle la version du serveur Apache. Pour ne pas permettre cette invitation gratuite à la corruption, nous avons ajouté la ligne suivante dans le fichier de configuration :

```
ServerSignature Off
```

En complément à cette mesure de sécurité nous avons défini notre page d'erreur 404 à la racine du site.

Si il y a une erreur 404 (document manquant), la page `Erreur404.html` est affichée. `ErrorDocument 404 /Erreur404.html`

9.3.2 Limiter les Denials of Service

Même s'il est vrai que les dénis de service sont moins probables avec un intranet qu'avec un Internet, il faut tout de même les prévenir : les attaques ne proviennent pas toujours que de l'extérieur... Aussi il est conseillé de limiter le nombre de connexions simultanées `MaxClients` et en particulier le nombre

de connexions persistantes `MaxKeepAliveRequests`. Celles-ci permettent d'effectuer des requêtes successives lors de la même connexion, ce qui augmente les performances du serveur. Enfin, l'utilisation d'un `Timeout` empêche les connexions sans fin. Avec un peu d'imagination, on pourrait avoir un parc un peu plus peuplé que celui que l'on a effectivement en salle de TP. Une configuration possible serait alors :

```
# 150 connexions simultanées maximum.
MaxClients 150
# On permet les connexions persistantes.
KeepAlive On
# 100 connexions persistantes simultanées.
MaxKeepAliveRequests 100
# Apache attendra 5 secondes après une connexion avant de la fermer.
KeepAliveTimeout 5
```

9.3.3 Définir correctement des Virtual Host

Le terme `Virtual Host` fait référence à la pratique de maintenir plus d'un seul serveur sur une même machine, différenciés par leur `hostname`. Par exemple, il est souvent avantageux pour des entreprises de partager un serveur web avec leurs noms de domaines respectifs accessibles sous les noms `www.companiel.com` et `www.companie2.com`.

Ici, la mesure de sécurité consiste à prévenir les pannes de serveur DNS ou des manipulations frauduleuses. Il convient pour cela de définir le `VirtualHost` par une adresse IP.

```
<VirtualHost 192.168.2.1>
  ServerName
</VirtualHost>
```

9.3.4 Traiter la politique de gestion des accès aux répertoires, aux fichiers et aux arborescences

Trois directives existent pour la gestion des accès aux répertoires, arborescence et fichiers du serveur. Ce sont respectivement : `Directory`, `Location` et `Files`.

Ces directives sont sous forme de balises englobantes dans lesquelles on peut définir les accès à ceux-ci.

Par exemple :

```
<Directory />
Order deny,allow
Deny from all
</Directory>
```

Cette directive est celle que nous avons utilisé pour notre serveur :

La ligne `Order` indique au serveur qu'il doit tout interdire par défaut à la racine du site, ensuite autoriser s'il y a lieu (avec l'instruction `AllowOverride`).

Ainsi il faudra définir des directives `AllowOverride` pour attribuer des droits d'accès aux répertoires apparents du site (la partie émergente de l'iceberg).

Une autre directive peut être placée dans les balises. Il s'agit de **Options**.

Options controle :

- le suivi des liens symboliques `FollowSymLinks/symIfOwnerMatch` que nous supprimons.

En effet, un pirate pouvant écrire dans un répertoire du serveur Web, par exemple via un serveur NFS, pourrait en profiter pour accéder au fichier `/etc/passwd` via un lien symbolique.

- l'exécution des scripts CGI, *ExceCGI* que nous n'utiliserons pas ;
- les Server Side *Includes* et *IncludesNOEXEC* que nous ne permettons pas non plus ;
- la génération de pages d'index *Indexes* en l'absence de celle-ci que nous allons désactiver. Nous verrons pourquoi.
- ainsi que l'orientation multilingue *Multiviews* qui ne nous est d'aucune utilité.

Désactiver la génération automatique d'index est un élément simple, mais important de la sécurité. En effet, si dans notre site, nous oublions de mettre un fichier nommé `index.html` dans un répertoire du site, alors une page indexant tous les fichiers présents dans celui-ci serait affichée.

Cela est non souhaitable car un pirate pourrait ainsi avoir connaissance de la présence de fichiers confidentiels.

Nous ajoutons donc la directive `Options`.

```
<Directory />
Order deny,allow
Deny from all
# Suppression des options.
Options None
</Directory>
```

9.3.5 Protection par mot de passe

Le module `mod_auth` permet de protéger l'accès à un répertoire par mot de passe. En pratique, c'est souvent utilisé pour filtrer l'accès à un répertoire déterminé.

Pour filtrer des répertoires personnels sur le site, les utilisateurs peuvent insérer dans le dossier dit un fichier `.htaccess` qui contient des directives (comme dans le fichier `httpd.conf`).

```
<Location "/prive">
# On n'autorise aucune option
Options None
# Aucune règle ne peut agir pour accéder autrement au répertoire /prive
AllowOverride None
AuthName "Espace Prive"
AuthType Basic
# Les mots de passe cryptés sont stockés dans le fichier /etc/httpd/.passwd
# Ce fichier se trouve en dehors du site.
AuthUserFile "/etc/httpd/.passwd"
require valid-user
</Location>
```

Le fichier `passwd` peut être créé par la commande `htpasswd` :

```
>htpasswd -cmb .passwd yoyo monpasswd
>htpasswd -mb .passwd jeff computer
...
```

Fichier `passwd` résultant :

```
yoyo:$apr1$NmWTM...$RunNlkdh2LLPCTrh/zEKm/
jeff:$apr1$1Bhuq/..$rbHAESAEkD45JJG/kLS5o0
```

9.4 Module PHP

Nous n'avons pas eu le temps de traiter correctement la partie sécurité de PHP. Aussi nous ne nous attarderons pas sur l'installation et tous les mécanismes de PHP. Ceci dit certaines versions de PHP contiennent de nombreuses failles de sécurité.

Toutefois voici ce que dit le manuel de PHP concernant la sécurité :

Le langage PHP a été pensé afin d'être un langage beaucoup plus sécurisé pour écrire des CGI que le Perl ou le langage C. De plus, une sélection rigoureuse des options de compilation et d'exécution vous permettra d'obtenir un équilibre parfait entre liberté et sécurité.

<http://php3.de/manual/fr/security.php>

Il est bien, comme pour tout système d'information, de venir aux nouvelles régulièrement afin d'appliquer certaines mesures correctives ou de mettre à jour la version de PHP.

Voici un lien à exploiter : <http://www.php.net/downloads.php>

Voici un autre document succinct qui montre quelles sont les directives de configurations relatives à la sécurité : http://www.phpteam.net/salon_php/secure/php_secure.pdf

Chapitre 10

Sécurité des mots de passe

10.1 Introduction

10.1.1 Utilité

Crack5 est un perceur de mot de passe.

Comme pour beaucoup d'outils de sécurité réseau, Crack est utilisé aussi bien par les crackers que par les administrateurs.

Il est utilisé par les administrateurs réseau pour **empêcher leurs utilisateurs de choisir des mots de passe trop simples à deviner**.

Crack est donc un outil nécessaire, vital même, à tout administrateur, puisque ne pas l'utiliser serait une brèche indirecte dans le système. En effet, n'importe quel cracker serait susceptible de deviner des mots de passe trop faciles. C'est donc pour cette raison que nous avons décidé de l'utiliser.

NB : Crack perce les mots de passe à partir d'un fichier de mots de passe tel que /etc/passwd. Il faut y avoir un accès (au moins en lecture) pour en tester le contenu. Ceci signifie que le cracker, s'il agit à distance, doit s'introduire par le moyen d'un telnet ou d'un ftp et doit donc connaître au moins un mot de passe utilisateur.

10.1.2 Domaine d'application et mode opératoire.

Crack est un utilitaire Unix, écrit pour Unix, tournant sur Unix.

10.2 Installation de Crack 5.0a

Après avoir dézippé crack5.0.tar.gz dans un répertoire approprié tel que /usr/local, le répertoire c50a est créé.

1. Tout d'abord il faut renommer quelques fichiers dans le dossier c50a/src/ :

```
>mv src/libdes src/libdes.orig
>cd src/util
>cp elcid.c,bsd elcid.c
```

Après être revenu dans le dossier c50a/, il faut éditer le script principal Crack.

Modifions la variable CRACK_PATH en :

```
CRACK_PATH=/usr/local/bin:usr/sbin:/sbin:/usr/bin:/bin:\$PATH
```

2. Ensuite, il faut mettre en commentaire les lignes concernant "vanilla" et décommenter les lignes du compilateur GCC GNU :

```
#
# now pick your compiler
#

# vanilla unix cc
#CC=cc
#CFLAGS="-g -O $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg: NetBSD MD5

# gcc 2.7.2
CC=gcc
CFLAGS="-g -O2 -Wall $C5FLAGS"
LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg: NetBSD MD5
```

3. Puis exécuter la commande de compilation du programme, puis du dictionnaire(cette dernière peut prendre plusieurs minutes).

```
>./Crack -makeonly
>./Crack -makedict
```

10.3 Exploitation de Crack 5

1. Ceci est suffisant pour utiliser le programme Crack5.0. Pour le lancer se placer dans le répertoire du script Crack et exécuter

```
>./Crack fichierPasswd
```

où fichierPasswd est un fichier de mots de passe Unix tel que /etc/passwd ou encore /etc/shadow.

2. Le programme rend la main assez rapidement, mais il ne faut pas croire qu'il en a finit. Au contraire, des programmes tournent en tâche de fond. La commande 'top' permet de le constater. De temps en temps, il faut regarder si des mots de passe ont été découverts. Pour cela, il y a le script ./Reporter qui est bien utile. Il faut l'exécuter pendant que Crack tourne. Un rapport est alors affiché à l'écran.

NB : Il est bien de rediriger la sortie standard vers un fichier :

```
>./Reporter > fichierRapport
```

Voici un exemple du résultat envoyé dans le fichier "fichierRapport" :

```
---- passwords cracked as of mar déc 10 10:10:13 CET 2002 ----
```

```
Gussed jeff [computer] ,,, [/etc/passwd /bin/bash]
Gussed root [<no-ciphertext>] Operateur b710pbx [/etc/passwd /bin/bash]
```

```
---- errors and warnings ----
```

```
ignoring locked entry: backup:*:34:34:backup:/var/backups:/bin/sh
ignoring locked entry: bin:*:2:2:bin:/bin:/bin/sh
ignoring locked entry: daemon:*:1:1:daemon:/usr/sbin:/bin/sh
ignoring locked entry: games:*:5:100:games:/usr/games:/bin/sh
ignoring locked entry: gnats:*:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats/gnats-db:/bin/sh
```

```

ignoring locked entry: irc:*:39:39:ircd:/var:/bin/sh
ignoring locked entry: list:*:38:38:SmartList:/var/list:/bin/sh
ignoring locked entry: lp:*:7:7:lp:/var/spool/lpd:/bin/sh
ignoring locked entry: mail:*:8:8:mail:/var/spool/mail:/bin/sh
ignoring locked entry: majordom:*:30:31:Majordomo:/usr/lib/majordomo:/bin/sh
ignoring locked entry: man:*:6:100:man:/var/cache/man:/bin/sh
ignoring locked entry: msql:*:36:36:Mini SQL Database Manager:/var/lib/msql:/bin/sh
ignoring locked entry: news:*:9:9:news:/var/spool/news:/bin/sh
ignoring locked entry: nobody:*:65534:65534:nobody:/home:/bin/sh
ignoring locked entry: operator:*:37:37:Operator:/var:/bin/sh
ignoring locked entry: postgres:*:31:32:postgres:/var/lib/postgres:/bin/sh
ignoring locked entry: proxy:*:13:13:proxy:/bin:/bin/sh
ignoring locked entry: sync:*:4:100:sync:/bin:/bin/sync
ignoring locked entry: sys:*:3:3:sys:/dev:/bin/sh
ignoring locked entry: uucp:*:10:10:uucp:/var/spool/uucp:/bin/sh
ignoring locked entry: www-data:*:33:33:www-data:/var/www:/bin/sh
StoreDataHook: invalid ciphertext: gdm !
StoreDataHook: invalid ciphertext: identd !
StoreDataHook: invalid ciphertext: telnetd !
StoreDataHook: wg='gdm Gnome Display Manager' un='gdm'
  cm='Gnome Display Manager [/etc/passwd /bin/false]' ct='!' sk='!'
StoreDataHook: wg='identd ' un='identd' cm=' [/etc/passwd /bin/false]' ct='!' sk='!'
StoreDataHook: wg='telnetd ' un='telnetd' cm=' [/etc/passwd /bin/false]' ct='!' sk='!'

---- done ----

```

3. Pour tuer le processus proprement, il faut exécuter :

```
>./script/plaster #(arret de la session de Crack)
```

et pour préparer une nouvelle session :

```
>./make tidy
```


Chapitre A

Types ICMP

Code	Type
echo-reply (pong)	
destination-unreachable	network-unreachable host-unreachable protocol-unreachable port-unreachable fragmentation-needed source-route-failed network-unknown host-unknown network-prohibited host-prohibited TOS-network-unreachable TOS-host-unreachable communication-prohibited host-precedence-violation precedence-cutoff
source-quench	
redirect	network-redirect host-redirect TOS-network-redirect TOS-host-redirect
echo-request (ping)	
router-advertisement	
router-solicitation	
time-exceeded (ttl-exceeded)	ttl-zero-during-transit ttl-zero-during-reassembly
parameter-problem	ip-header-bad required-option-missing
timestamp-request	
timestamp-reply	
address-mask-request	
address-mask-reply	

TAB. A.1 – Liste des types ICMP gérés par ipchains 1.3.9, 17-Mar-1999

Chapitre B

Syntaxe log ipchains

Voici une ligne de notre fichier de log `/etc/messages`.

```
Nov  8 10:25:53 b710pbv kernel: Packet log: input REJECT eth1 PROTO=17 134.214.90.16:138
134.214.91.255:138 L=236 S=0x00 I=51490 F=0x0000 T=128 (\#7)
```

Exemple	Description
Nov 8	date
10 :25 :53	heure
b710pbv	nom de la machine
kernel :	daemon
Packet log :	précède le contenu du paquet logué
input	nom de la chaîne traversée par le paquet
REJECT	nom de la police affectée à la chaîne
eth1	nom de l'interface tracversée le paquet
PROTO=17	numéro du protocole
134.214.90.16 :138	adresse IP source et port (TCP ou UDP) ou type ICMP
134.214.91.255 :138	adresse IP destination et port (TCP ou UDP) ou type ICMP
L=236	taille totale du paquet
S=0x00	TOS (Type Of Service)
I=51490	identifiant numérique unique du paquet IP
F=0x0000	Drapeau (3 bits) et numéro de fragment (13 bits)
T=128	TTL (Time To Live)
(\#7)	numéro de la règle ipchains qui a causé la mise en log

TAB. B.1 – Syntaxe du fichier de log

Table des matières

1	Introduction	3
1.1	Sujet	3
1.2	Limitations	3
1.3	Répartition des tâches	3
1.4	Présentation orale	4
2	Architecture	5
3	Firewall	7
3.1	L'outil ipchains	7
3.1.1	Présentation	7
3.1.2	Fonctionnement	7
3.1.3	Critères de filtrage	7
3.1.4	Actions possibles sur les paquets	8
3.1.5	Mode de construction des chaînes et des règles	8
3.2	Stratégie	8
3.2.1	Réseau local invisible	9
3.3	Passerelle sécurisée	9
3.3.1	Initialisation des règles de filtrage	9
3.3.2	Accès http	10
3.3.3	Processus locaux	10
3.3.4	Réseau privé	10
3.3.5	anti - IP Spoofing	11
3.3.6	Restrictions telnet	11
3.3.7	Trafic IP	11
3.3.8	Trafic ICMP	11
3.3.9	Fichier de log	12
3.4	Testons le service ftp	13
3.5	Testons le service telnet	13
3.5.1	Connexion	13
3.5.2	Navigation	14
3.5.3	Accès aux périphériques	14
3.5.4	Droits super utilisateur	15
3.5.5	Déconnexion	15
3.6	SSH	15

4	SSH	17
4.1	Introduction	17
4.2	Pourquoi utiliser SSH?	17
4.3	Introduction à la cryptographie	17
4.3.1	Le cryptage symétrique :	17
4.3.2	Le cryptage asymétrique :	18
4.4	Scénario d'usage	19
4.5	Exiger SSH pour les connexions à distance	19
4.6	Sources	19
4.7	Fichiers de configuration d'OpenSSH	20
4.7.1	Le fichier <code>ssh_config</code> :	20
4.7.2	Le fichier <code>sshd_config</code> :	21
4.7.3	Un fichier très simple de <code>sshd_config</code> :	23
4.8	Installation	23
4.9	Configuration	23
4.9.1	Client	23
4.9.2	Serveur	24
4.10	Testons la connexion	24
4.10.1	Copie (utilisation de la commande <code>scp</code>) :	25
4.10.2	Tests avec <code>Ethereal</code>	25
4.10.3	Déconnexion	25
4.11	Exemple réel de session SSH	25
5	Scanning	27
5.1	Présentation	27
5.1.1	Installation	27
5.1.2	Description	27
5.2	Les résultats de <code>NMAP</code>	28
5.2.1	Première prise en main	28
5.2.2	Test sur un routeur autorisant des entrées	29
5.2.3	Cacher l'adresse d'envoi des requêtes	30
5.2.4	Scanner un réseau	30
6	Rootkits et Trojans	33
6.1	Définitions	33
6.2	Résumé : étapes du hacker	33
6.3	Mon expérimentation	34
6.4	Comment se présente le programme?	34
6.5	Mode de fonctionnement	34
6.6	Top du moment	35
6.7	Les protections existantes	35
7	Audit de réseau avec <code>nessus</code>	37
7.1	Introduction	37
7.2	Sources	37
7.3	Installation	37
7.4	Configuration	38
7.5	Utilisation	38

7.6	Tests effectués et résultats observés	38
7.6.1	Test sur une machine protégée totalement	48
7.6.2	Test sur une machine laissant peu de ports accessibles	48
7.6.3	Test sur un serveur web	49
7.7	Conclusion	49
8	Forger des paquets avec μito	51
8.1	Introduction	51
8.2	Installation	51
8.3	Utilisation et test	51
8.3.1	Utilisation	51
8.3.2	Test	54
8.4	Conclusion	54
9	Intranet : Serveur HTTP Apache	55
9.1	Choix du serveur	55
9.2	Installation	55
9.3	Configuration du serveur	56
9.3.1	Cacher les bannières	56
9.3.2	Limiter les Denials of Service	56
9.3.3	Définir correctement des Virtual Host	57
9.3.4	Traiter la politique de gestion des accès aux répertoires, aux fichiers et aux arborescences	57
9.3.5	Protection par mot de passe	58
9.4	Module PHP	59
10	Sécurité des mots de passe	61
10.1	Introduction	61
10.1.1	Utilité	61
10.1.2	Domaine d'application et mode opératoire	61
10.2	Installation de Crack 5.0a	61
10.3	Exploitation de Crack 5	62
A	Types ICMP	65
B	Syntaxe log ipchains	67
	Bibliographie	71

Bibliographie

- [1] Ambrose Au. *Linux IP Masquerade mini HOWTO*.
<http://www.freenix.fr/unix/linux/HOWTO/mini/IP-Masquerade.html>.
- [2] Manfred Bartz. *Ipchains Log Format*.
<http://logi.cc/linux/ipchains-log-format.php3>.
- [3] Emmanuel Cecchet. *La pile TCP/IP*.
<http://sardes.inrialpes.fr/people/cecchet/teaching/Cours2.ppt>.
- [4] Maes Dominique. *La Sécurité sur NT*.
<http://users.win.be/W025042/securnt/dico.htm>.
- [5] S. Ventura et Biasino Cassella. *Présentation sur le tunneling utilisant SSL, SSH, IPsec*.
<http://www.tcom.ch/Presentations/Tunnel/tunnel.htm>.
- [6] Gilbert LE HUU HOA et Paul SUCH. *Projet GLIPS*.
http://www.epita.net/projets/projets_details_43.htm.
- [7] Mark Grennan. *Le HOWTO du pare-feu et des serveurs mandataires*.
<http://www.freenix.fr/unix/linux/HOWTO/Firewall-HOWTO.html>.
- [8] Olivier Hoarau. *Auditer la sécurité de son réseau*.
<http://funix.chez.tiscali.fr/informatique/linux/audit-secu.htm>.
- [9] Olivier Hoarau. *Configurer un firewall avec ipchains*.
<http://funix.chez.tiscali.fr/informatique/linux/ipchains.htm>.
- [10] Nicolas JUSTIN. *Étude sur les Port Scans et Élaboration d'un Outil de Détection*.
<http://nicolas.justin.free.fr/TCPScan/TCPScan.html>.
- [11] Francois Laissus. *Cours d'introduction à TCP/IP, Format des messages ICMP*.
<http://www.laissus.fr/cours/node71.html>.
- [12] Amaury DUCHESNE et Phat PIRON Laurent BARSIN, Olivier DELCOURT. *SEMINAIRE DE DETECTION D'INTRUSIONS, Les différentes techniques de scans et outils de détection*.
http://www.student.montefiore.ulg.ac.be/barsin/detection/portscan_report_1704.html/portscan_report.html.
- [13] Toby Miller. *Analyse du Root Kit Knark*.
<http://ouah.sysdoor.net/RootkitKNARKfr.htm>.
- [14] NewWare. *Sécurité des systèmes et des communications*.
<http://newwar.free.fr/netbios.htm>.
- [15] Pascal Nicolas. *Cours de réseaux, Le protocole IP, La gestion des erreurs*.
<http://www.info.univ-angers.fr/pub/pn/poly/node42.html>.
- [16] Bernard Perrot. *Root-kit et intégrité*.
<http://www.security-labs.org/index.php3?page=404>.

- [17] Bernard Perrot. *Sécuriser ses connexions avec ssh*.
<http://www.security-labs.org/index.php3?page=405>.
- [18] Pierre Plet. *Réunion RESIST (Réseaux et Systèmes d'Information Sécurisés à Toulouse) du 27 mai 2002-05-27*.
<http://www.ossir.org/resist/cr/200205/20020527.html>.
- [19] Paul Russell. *Linux IPCHAINS-HOWTO*.
<http://www.freenix.fr/unix/linux/HOWTO/IPCHAINS-HOWTO.html>.
- [20] Jean Saquet. *Protocoles d'applications*.
<http://users.info.unicaen.fr/~jean/NAPI/cours2.html>.