

Un petit guide pour la sécurité

Alexandre Viardin
Mirabellug

guidesecu(at)free.fr

Publié par
Philippe Latu



Un petit guide pour la sécurité

Publié par et Alexandre Viardinet Philippe Latu

Copyright et Licence

Copyright (c) 2003 Alexandre Viardin
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Copyright (c) 2003 Alexandre Viardin
Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU (GNU Free Documentation License), version 1.1 ou toute version ultérieure publiée par la Free Software Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture, et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est incluse dans la section intitulée « Licence de Documentation Libre GNU ».

Historique des versions

Version .Revision: 1.7 . .Date: 2004/01/04 20:21:18 . Revu par : pl

Publication Linux France

Version \$Revision: 1.10 \$ \$Date: 2004/02/17 21:25:38 \$ Revu par : pl

Compléments sur le Peer To Peer et les virus.

Table des matières

Avant-propos	i
1. Pourquoi ce guide ?.....	i
2. Où trouver ce guide ?.....	i
3. Quels sont les systèmes d'exploitation visés ?.....	ii
1. Sécurisation de base	1
1.1. Premier conseil : Verrouillez les stations.....	1
1.2. Pour Linux.....	1
1.3. Pour Windows.....	1
1.4. Le lecteur de disquettes.....	1
1.5. Le lecteur de CDROM.....	1
1.6. N'oubliez pas le mot de passe pour le BIOS.....	1
2. La collecte d'informations	3
2.1. Le Scanner.....	3
2.1.1. Qu'est ce qu'un scanner ?.....	3
2.1.2. Comment marche Nmap ?.....	4
2.1.3. La détermination du système d'exploitation avec Nmap.....	5
2.1.4. Quel est l'intérêt d'utiliser Nmap ?.....	6
2.1.5. Comment s'en protéger ?.....	6
2.1.6. Documents.....	6
2.2. Identifier les versions des logiciels en écoute.....	6
2.2.1. Netcat.....	7
3. Les failles applicatives	8
3.1. Les installations par défaut.....	8
3.2. Les mauvaises configurations.....	8
3.3. Les bogues.....	8
3.3.1. Des dénis de services applicatifs.....	8
3.3.2. Outrepassement de droits.....	8
3.3.3. Les scripts.....	8
3.4. Les exploits.....	9
3.5. Comment s'en protéger ?.....	9
4. Les outils indispensables pour la protection	11
4.1. Le pare-feu <i>firewall</i>	11
4.1.1. La configuration.....	11
4.1.2. Les attaques contre les firewalls.....	11
4.2. Les systèmes de détection d'intrusion (HIDS/NIDS).....	12
4.2.1. Prelude-NIDS.....	13
4.2.2. Snort.....	13
4.3. Le tunneling.....	13
4.3.1. Le protocole AH.....	13
4.3.2. Le protocole ESP.....	13
4.3.3. Le protocole IPcomp.....	14
4.3.4. Le protocole IKE.....	14
4.3.5. Les deux modes de fonctionnements de IPsec.....	14
4.3.6. Les limitations d'IPsec.....	14
4.3.7. Documents.....	14
4.4. Nessus.....	14
4.4.1. Pour obtenir tout sur Nessus.....	14
4.5. User Mode Linux - UML.....	14
4.5.1. Documents.....	15

5. Surveillance - Dissimulation - Maintien d'accès	16
5.1. Les chevaux de Troie	16
5.1.1. Comment s'en protéger ?	16
5.2. Les backdoors	16
5.2.1. Les backdoors présentes dans les logiciels	16
5.2.2. Les backdoors dédiées aux connexions à distance	17
5.3. Les Rootkits	17
5.3.1. Comment s'en protéger ?	17
5.4. L'interception des mots de passe en réseau	18
5.4.1. Comment s'en protéger ?	18
6. Dispositifs destructeurs	20
6.1. Le virus	20
6.1.1. Comment s'en protéger ?	20
6.2. Les vers	20
6.2.1. Comment s'en protéger ?	20
6.3. Les bombes logiques	21
6.3.1. Comment s'en protéger ?	21
6.4. Les attaques par déni de services	21
6.4.1. Le SYN flood	21
6.4.2. L'UDP Flood	21
6.4.3. La fragmentation de paquets	22
6.4.4. Ping of death	22
6.4.5. Attaque par réflexion : Smurfing	22
6.4.6. Dénis de services distribués	22
6.4.7. Bombes e-mail	22
7. Sécurisation des mots de passe	24
7.1. L'attaque par dictionnaire	24
7.2. Le brute forcing	24
7.3. Tester la fiabilité de vos mots de passe !	24
7.4. Choisir le bon mot de passe	25
7.5. Prévenir l'utilisateur	25
8. La base des attaques réseaux	26
8.1. Détournement de flux	26
8.1.1. ARP-Poisoning	26
8.1.2. Désynchronisation TCP	27
8.2. <i>Man In the Middle</i> - MITM	27
8.2.1. Document	28
8.3. Encapsulation d'IP dans d'autres protocoles	28
9. Description d'attaques sur différents protocoles	29
9.1. Dynamic Host Configuration Protocol - DHCP	29
9.1.1. Attaque par épuisement de ressources	29
9.1.2. Faux serveurs DHCP	29
9.1.3. Comment s'en protéger ?	29
9.1.4. Documents	30
9.2. Domain Name Service - DNS	30
9.2.1. Le DNS ID spoofing	30
9.2.2. Le DNS cache poisoning	31
9.2.3. Comment s'en protéger ?	32
9.2.4. Documents	32
9.3. FINGER	32
9.3.1. Comment s'en protéger ?	33
9.4. FTP	33
9.4.1. Le serveur FTP anonyme	33
9.4.2. Comment s'en protéger ?	34
9.5. HTTP	34
9.5.1. Les serveurs trop bavards	34
9.5.2. Vulnérabilités liées aux applications web	34
9.5.3. Comment se protéger ?	35

9.6. IDENT	35
9.6.1. Comment s'en protéger ?	35
9.7. IP et l'IP-Spoofing	35
9.7.1. Un peu de théorie	36
9.7.2. Prévenir l'IP spoofing grâce à Nmap	38
9.7.3. Comment s'en protéger ?	38
9.7.4. Document	39
9.8. NETBIOS	39
9.8.1. Comment s'en protéger ?	39
9.8.2. Document	39
9.9. Network File System - NFS	39
9.9.1. Les attaques	39
9.9.2. Comment s'en protéger ?	39
9.10. Network Information Service - NIS	39
9.10.1. Les attaques	40
9.10.2. Comment s'en protéger ?	40
9.11. PORTMAP	40
9.11.1. Comment s'en protéger ?	40
9.12. Le protocole SMB	40
9.12.1. Les scans de SMB shares	41
9.12.2. Comment s'en protéger ?	41
9.12.3. Document	41
9.13. Le service de messagerie - SMTP	41
9.13.1. Comment s'en protéger ?	42
9.14. SQL	42
9.14.1. L'injection SQL ou SQL-Injection	42
9.14.2. Comment s'en protéger ?	42
9.14.3. Document	42
9.15. SSH	42
9.16. TELNET	43
9.16.1. Comment s'en protéger ?	43
9.17. XWINDOW	43
9.17.1. Les attaques	43
9.17.2. Comment s'en protéger ?	43
9.18. Peer To Peer (eDonkey, Kazaa, etc.)	43
9.18.1. Les outils Peer To Peer sont des vecteurs de virus	44
9.18.2. Comment s'en protéger ?	44
10. Sécurité avancée	45
10.1. L'architecture sécurisée	45
10.1.1. Le réseau de départ	45
10.1.2. Le premier niveau de sécurité	45
10.1.3. Le deuxième niveau de sécurisation	46
10.1.4. Les niveaux plus élevés	46
10.2. Développez vos propres utilitaires sécurité	47
10.2.1. Le programme	48
10.2.2. Comment obtenir et compiler le source entier du programme ?	52
10.2.3. Documents	52
A. Annexes	53
A.1. Les sites et revues à consulter régulièrement	53
A.2. Remerciements	53

Avant-propos

«Qui connaît l'autre et se connaît, en cent combats ne sera point défait; qui ne connaît l'autre mais se connaît, sera vainqueur une fois sur deux; qui ne connaît pas plus l'autre qu'il ne se connaît sera toujours défait.»

L'art de la guerre - Sun Tzu

1. Pourquoi ce guide ?

Ce guide a été réalisé suite à un audit de sécurité que j'ai réalisé pour mon école et aux deux conférences sur la sécurité réseau présentées au groupe d'utilisateurs Linux de NANCY (coucou le Mirabellug). Je ne suis pas spécialiste en sécurité réseau ; j'ai juste écrit ce guide dans le but de donner à des administrateurs ou à des particuliers, un descriptif technique et un manuel d'autoformation à la sécurité réseau.

La plupart des administrateurs ne sont pas spécialistes en sécurité, et peuvent être perdus devant un problème de ce type. Le masse d'informations disponible sur Internet est parfois confuse, dense ou très technique. Ce guide sert de point de départ et d'introduction à la sécurité. Il a été pensé dans un but évolutif. Si vous voulez participer en écrivant ou en complétant des chapitres, n'hésitez pas à me contacter à l'adresse `guidesecu(at)free.fr`.

Le principe est simple : une description assez succincte sur une attaque et ensuite une description complète des méthodes de protection. Vos expériences personnelles et vos remarques sont aussi les bienvenues. Evidemment, ce guide est sous license GFDL donc gratuit. La seule récompense que pourront recevoir les éventuels participants est la mention de leurs noms en tant que collaborateurs.

Ce guide se compose d'une dizaines de chapitres. Chaque chapitre comporte une introduction. La plupart du temps, un chapitre contient au moins une section divisée en différentes sous sections : une pour la description d'un problème de sécurité particulier, une deuxième pour décrire les différents moyens de s'en protéger et une troisième pour donner différents liens vers des documents plus précis sur le sujet.

Le premier chapitre montre comment sécuriser une station pour éviter toutes tentatives de piratage par un accès physique.

Le deuxième chapitre décrit le fonctionnement des outils de récupération d'informations à distance, notamment les scanners. Il montre l'utilité qu'ils ont pour vous protéger.

Le troisième chapitre introduit la notion de failles.

Le quatrième chapitre introduit différentes notions sur les firewalls et les principaux autres systèmes de protection logiciels.

Le cinquième chapitre explique comment un pirate dissimule sa présence sur un système.

Le sixième chapitre s'intéresse aux dispositifs destructeurs (virus, bombes mails, ...).

Le septième chapitre décrit les attaques sur les fichiers de mots de passe.

Les huitième et neuvième chapitres traitent de différents problèmes posés par certains protocoles réseaux.

Le dixième chapitre est divisé en deux parties : la première explique comment architecturer son réseau de façon sécurisée. La deuxième partie est un cours sur le développement d'outils dédiés uniquement à la sécurité.

2. Où trouver ce guide ?

C'est très simple, il y a plusieurs adresses :

- Le site officiel : <http://guidesecu.ifrance.com/guide/guidesecu.pdf>
- Sur le site du Mirabellug : <http://www.mirabellug.org/docs/securite/guidesecu.pdf>
- Le site Linux France : <http://www.linux-france.org/prj/inetdoc/securite/tutoriel/>

3. Quels sont les systèmes d'exploitation visés ?

La majorité des programmes défensifs utilisés et décrits dans ce guide sont disponibles sous LINUX. Je n'oublierai pas de parler de la sécurité pour les produits Microsoft. Cependant, Linux possède une certaine avance sur Microsoft dans le domaine de la sécurité (notamment par un plus grand nombre de logiciels performants et gratuits).

Les autres systèmes comme SunOS, VMS, MacOS, Plan9, Novell,... seront passés sous silence mais si vous voulez voir des chapitres précis sur certains OS apparaitre, contactez moi par mail.

Bonne Lecture !

Chapitre 1. Sécurisation de base

Introduction

Le but de ce chapitre est de donner différentes méthodes pour sécuriser physiquement une machine. Il faut savoir qu'une grande partie des piratages sont lancés par des pirates ayant un accès physique sur un réseau.

Dans ce chapitre, nous ne nous focaliserons pas sur un serveur dédié à un service particulier, mais plutôt sur les machines constituant les clients. Ces machines sont en accès libre dans une salle non surveillée.

L'objectif est d'empêcher une personne mal intentionnée d'obtenir les accès administrateur sur la machine qu'elle utilise. La plupart des utilitaires de piratage ont besoin des accès administrateur pour fonctionner ; sans ces accès, la capacité de nuire est fortement diminuée.

1.1. Premier conseil : Verrouillez les stations

N'hésitez pas à poser un cadenas sur les tours des machines, cela empêchera tout d'abord le vol de matériel, mais cela évitera aussi d'avoir des disques durs montés en «secret» avec toute une panoplie d'utilitaires installés dessus. Le conseil à suivre impérativement (et vous comprendrez pourquoi en lisant les deux chapitres suivants) : il faut désactiver le boot sur le lecteur de disquette et sur le lecteur de CDROM.

1.2. Pour Linux

Evitez d'avoir l'option `failsafe` au démarrage proposé par Lilo. Cette option peut permettre d'obtenir les accès root (sans mot de passe) pour la maintenance du système.

1.3. Pour Windows

Le système de fichier NTFS permet une sécurisation accrue par rapport aux systèmes de fichier FAT et FAT 32. Si vos machines Windows fonctionnent avec un système FAT, passez en NTFS. Je déconseille fortement d'utiliser Windows 95, 98 et Me, le niveau de sécurité offert par ces systèmes en natif n'étant pas assez élevé.

1.4. Le lecteur de disquettes

Evitez le boot sur disquette (certaines versions Linux s'installent en RAM grâce à un nombre limité de disquettes) qui donne la possibilité de monter tous les systèmes de fichiers présents sur le(s) disque(s) dur(s) de la machine et d'en modifier le(s) contenu(s). De plus, *Trinux* (<http://www.trinux.org>) est livré avec un panel assez impressionnant d'utilitaires exclusivement dédiés à la sécurité. Le programme NTFS2DOS (sous DOS) permet de changer les partitions NTFS en partitions FAT et de pouvoir accéder à leurs contenus sans restrictions. NTFS2DOS est lancé depuis une disquette de boot DOS.

1.5. Le lecteur de CDROM

Des utilitaires comme *Knoppix* (<http://www.knoppixfr.org>) (système Linux bootant sur un seul CD et contenant lui aussi un nombre impressionnant d'utilitaires divers) peuvent être utilisés pour monter les différents systèmes de fichiers présents sur le(s) disque(s) dur(s).

1.6. N'oubliez pas le mot de passe pour le BIOS

N'oubliez de protéger l'accès du BIOS par un mot de passe ! Attention certains BIOS peuvent comporter des failles logicielles permettant d'outrepasser ces protections. Encore une fois, il ne faut pas oublier de cadenasser les tours, afin d'éviter à des utilisateurs (encore) mal intentionnés de retirer la pile du BIOS et d'outrepasser la protection par mot de passe.

Chapitre 2. La collecte d'informations

Introduction

Dans ce chapitre, nous allons décrire le fonctionnement des outils permettant de récupérer des informations à distance. Ces utilitaires sont fréquemment utilisés par les pirates pour préparer de futures attaques. C'est pour cette raison qu'il est indispensable de les décrire dès le début. Vous apprendrez également à les utiliser pour votre propre protection.

2.1. Le Scanner

L'objectif du pirate est de repérer les serveurs offrant des services particuliers et de les identifier. Pour obtenir ces informations, le pirate va utiliser un *scanner*. Le but de ce section est de présenter des méthodes de protections contre le scan (en utilisant des règles de firewalling sous iptables/ipchains par exemple) et de savoir utiliser un *scanner* pour anticiper les futures attaques. Le *scanner* décrit dans ce chapitre est *Nmap* (<http://www.insecure.org>), un des *scanners* les plus utilisés et un des plus performants. *Nmap* est disponible sous Windows et Linux en paquetage dans toutes les distributions majeures. La version décrite dans ce chapitre étant celle disponible sous Linux. Je décrirai dans une première partie ce qu'est un scanner. Ensuite, je me focaliserai sur *Nmap* et je le présenterai d'un point de vue un peu plus technique, permettant de comprendre les différentes méthodes de protection.

Note : Attention : pour une capacité optimale de fonctionnement, *Nmap* doit être utilisé avec les droits du super-utilisateur root !.

2.1.1. Qu'est ce qu'un scanner ?

C'est très simple : lorsqu'un serveur offre un service particulier (Web, messagerie, mail), il exécute un programme assurant ce service. Ce programme est en attente de connexions. Les clients devant accéder à ce service doivent connaître l'adresse IP du serveur et le numéro de port associé au service. Ce numéro de port a été attribué suivant le document standard RFC1010 (<http://www.faqs.org/rfcs/rfc1010.html>) au programme exécutant ce service. Sur les systèmes Linux et *BSD la liste de ces numéros est disponible dans le fichier `/etc/services`. La plupart des services ont un numéro de port bien défini. Par exemple, un serveur de messagerie utilise le port 25, un serveur Web le port 80... Lorsqu'un service est en écoute sur un port, on dit que le numéro de port associé à ce service est ouvert. L'intérêt du *scanner* est très simple : il permet de trouver dans un délai très court, tous les ports ouverts sur une machine distante. Il existe différents types de *scanner*, certains se contentent juste de donner : la liste des ports ouverts, le type et la version de l'OS tournant sur le serveur (ces fonctionnalités seront décrites dans ce chapitre avec *Nmap*). D'autres *scanners* comme *Nessus* (<http://www.nessus.org>) permettent de tester différentes failles connues sur ces services. Voir [Section 4.4](#).

2.1.1.1. Exemple avec Nmap

Utilisons *Nmap* pour connaître les services en écoute sur la machine d'adresse IP 192.168.1.1 :

```
[root@nowhere.net /root]# nmap 192.168.1.1

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.1.1) :
(The 1544 ports scanned but not shown below are in state : closed)
Port State Service
21/tcp open ftp
53/tcp open domain
80/tcp open http
110/tcp open pop-3
111/tcp open sunrpc
```

```
113/tcp open auth
631/tcp open cups
845/tcp open unknown
901/tcp open samba-swat
10000/tcp open snet-sensor-mgmt
```

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds.

Nmap donne un aperçu assez complet des différents services s'exécutant sur la machine dans un temps assez bref.

On peut observer ici que des serveurs FTP, DNS, WEB, POP-3 ... sont en attente de connexions.

2.1.2. Comment marche Nmap ?

Je présenterai de manière très succincte Nmap et me focaliserai principalement sur les fonctions les plus utilisées.

Pour connaître les ports ouverts sur une machine, Nmap procède à l'envoi de paquets sur tous les ports de cette machine et analyse les réponses. Bien sûr, il y a différents types de scans, donc différents types d'envois et donc, différents types de réponses.

Nous nous intéresserons aux scans utilisant le protocole TCP (les scans UDP et ICMP étant possibles eux aussi).

2.1.2.1. Le scan *vanilla TCP connect*

Nmap procède à l'appel de la fonction connect() sur tous les ports de la machine. Ce type de scan est facilement repérable.

Le scan en *vanilla TCP connect* est le scan par défaut avec Nmap, la commande est :

```
[root@nowhere.net /root]# nmap [ip de la machine cible]
```

ou

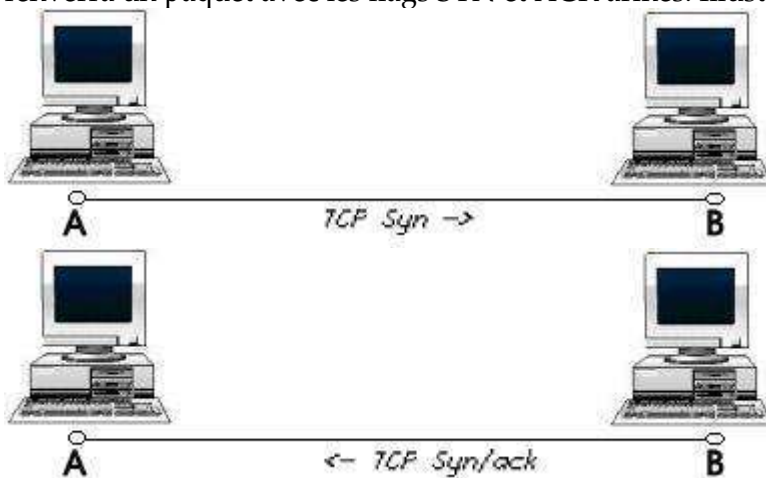
```
[root@nowhere.net /root]# nmap -sT [ip de la machine cible]
```

2.1.2.2. Les scans furtifs

Nous rentrons maintenant dans une classe de scans plus difficiles à détecter :

Le scan en connexion demi-ouverte ou "Syn-scan"

Nmap envoie sur chaque port un paquet TCP avec le flag SYN armé ; si un port est ouvert, il renverra un paquet avec les flags SYN et ACK armés. Illustration :



La commande se fait par l'appel de nmap avec l'option `-sS` :

```
[root@nowhere.net /root]# nmap -sS [adresse IP de la machine cible]
```

Les scans Xmas, FIN et NULL

Le scan FIN consiste en l'envoi de paquets TCP avec seulement le flag FIN armé. La commande se fait par l'appel de nmap avec l'option `-sF` :

```
[root@nowhere.net /root]# nmap -sF [adresse IP de la machine cible]
```

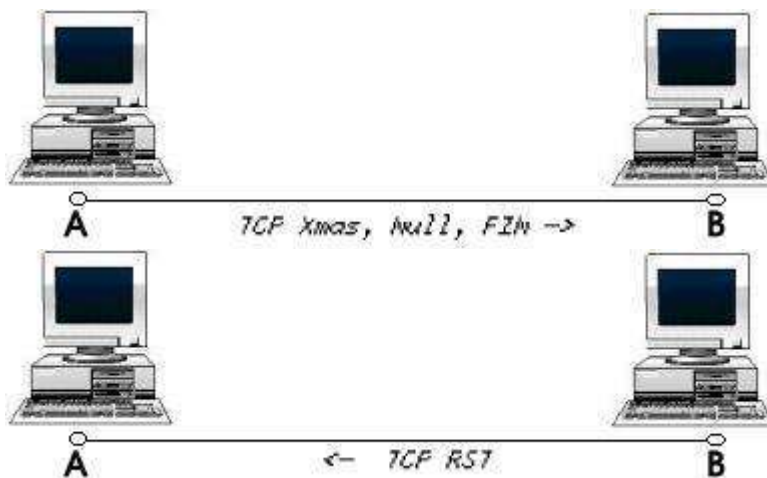
Le scan NULL consiste en l'envoi de paquets TCP avec seulement le flag NULL armé. La commande se fait par l'appel de nmap avec l'option `-sN` :

```
[root@nowhere.net /root]# nmap -sN [adresse IP de la machine cible]
```

Le Xmas scan (traduisez le scan de Noël) consiste en l'envoi de paquets TCP avec les flags FIN/URG/PUSH armés. La commande se fait par l'appel de nmap avec l'option `-sX` :

```
[root@nowhere.net /root]# nmap -sX [adresse IP de la machine cible]
```

Pour ces trois types de scans, les systèmes répondent avec un paquet RST si le port est fermé et ne répondent pas si le port est ouvert. Le NULL scan ne fonctionne pas contre des plateformes Microsoft. Illustration :



2.1.3. La détermination du système d'exploitation avec Nmap

Si on lance Nmap avec l'option `-O` :

```
[root@nowhere.net /root]# nmap -O 192.168.0.1

Starting nmap 3.48 ( http://www.insecure.org/nmap/ )
Interesting ports on (192.168.0.1):
(The 1647 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
113/tcp   open  auth
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
515/tcp   open  printer
587/tcp   open  submission
901/tcp   open  samba-swat
Device type: general purpose
Running: Linux 2.4.X (1)
OS details: Linux 2.4.20 - 2.4.21 w/grsecurity.org patch
Uptime 76.872 days (since Tue Sep  2 15:20:23 2003)

Nmap run completed -- 1 IP address (1 host up) scanned in 7.030 seconds
```

(1) Notez bien cette ligne : `Linux 2.4.X`.

Nmap parvient à déterminer le système d'exploitation tournant sur la machine cible. La machine cible utilise un noyau Linux 2.4.21-grsec. Nmap ne s'est pas trompé.

Il faut savoir que chaque système d'exploitation construit ses paquets d'une manière bien particulière. Certains champs au niveau de la couche IP ou TCP sont propres à chaque système d'exploitation. Nmap contient une base de données d'un grand nombre de systèmes. Nmap envoie donc des paquets tests à la machine cible et compare les paquets reçus en réponse à ceux de sa base de données et en déduit le type de système.

Cette base de données est mise à jour en fonction des différentes version de Nmap.

2.1.4. Quel est l'intérêt d'utiliser Nmap ?

Nmap permet de pouvoir prévoir les futures attaques, et aussi de pouvoir connaître quels services tournent sur une machine. Une installation faite un peu trop vite peut laisser des services en écoute (donc des ports ouverts sans que cela ne soit nécessaire) et donc vulnérables à une attaque. N'hésitez pas à utiliser Nmap contre vos serveurs pour savoir quels ports sont en écoute.

Nmap est un logiciel très complet et très évolutif, et il est une référence dans le domaine du *scanning*.

2.1.5. Comment s'en protéger ?

Configurer votre pare-feu pour empêcher les scans :

```
[root@nowhere /root]# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/3
```

Cette commande permet de détecter l'envoi un grand nombre de paquets TCP avec les flags FIN et/ou SYN et/ou ACK et/ou RST armé(s). Vérifiez que votre pare-feu (si ce n'est pas iptables) supporte la détection de scans.

2.1.6. Documents

- Je vous conseille de lire *Utiliser iptables :Spécifications de filtrage* (<http://www.linux-france.org/prj/inetdoc/guides/packet-filtering-HOWTO/packet-filtering-HOWTO-7.html#ss7.3>).
- Divers articles écrits par le développeur de Nmap sur le *scanning* (en anglais) : *The Art of Port Scanning* (<http://www.phrack.org/show.php?p=51&a=11>), *Remote OS detection via TCP/IP Stack FingerPrinting* (<http://www.phrack.org/show.php?p=54&a=9>) et *ICMP based remote OS TCP/IP stack fingerprinting techniques* (<http://www.phrack.org/show.php?p=57&a=7>).

2.2. Identifier les versions des logiciels en écoute

Maintenant que notre pirate connaît les différents services en écoute, son objectif va être de découvrir les noms et les versions des logiciels utilisés pour assurer ces services.

L'objectif de ce chapitre est de présenter une parade pour éviter de donner trop d'informations sur votre système.

Le pirate peut déjà essayer de se connecter sur différents ports grâce aux programmes clients associés pour glaner des informations. Rien que telnet donne beaucoup d'informations :

```
[root@nowhere.net /root]# telnet 192.168.1.1
```

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
```

```
Escape character is "CTRL-C".
Welcome to muetdhiver
Linux Mandrake release 7.2 (Odyssey) for i586
Kernel 2.2.17-21mdk on an i586
login :
```

Avec la réponse :

```
1,0,0Linux Mandrake release 7.2 (Odyssey) for i586
Kernel 2.2.17-21mdk on an i586
```

Trop de renseignements apparaissent.

Même si le port telnet est fermé, le pirate peut glaner d'autres informations sur les autres services. Pour cela, il peut procéder à une connexion telnet sur le port associé à un autre service en écoute. Exemple de connexion telnet sur le port FTP (port 21) :

```
[root@nowhere.net /root]# telnet 192.168.1.1 21

Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is 'CTRL-C'.
220 ProFTPD 1.2.5rc1 Server (ProFTPD Default Installation) [neuromancer]
```

La ligne qui nous intéresse le plus est celle-ci :

```
220 ProFTPD 1.2.5rc1 Server (ProFTPD Default Installation) [neuromancer]
```

La version du logiciel nous est donnée. Le pirate va alors rechercher des informations sur les faiblesses de sécurité de celui-ci.

Cependant, sur certains services, le client telnet sera inefficace. Le pirate peut alors utiliser un programme conçu pour écrire et lire de données sur machine cible et sur le port voulu. Ces programmes permettent d'envoyer des scripts directement sur le logiciel souhaité sans se soucier des limites protocolaires.

Le plus réputé de ces programmes est sans doute *Netcat*.

2.2.1. Netcat

Netcat permet d'établir une connexion (TCP ou UDP) sur un port souhaité et d'y envoyer ou d'y recevoir des données. Voici un exemple :

```
[root@nowhere.net /root]# nc 192.168.1.1 21

220 ProFTPD 1.2.5rc1 Server (ProFTPD Default Installation) [neuromancer]
```

On obtient directement la version du logiciel utilisé.

Netcat (<http://packetstormsecurity.nl/UNIX/netcat/>) comporte plein d'autres fonctionnalités (comme l'envoi de scripts ...). Le pirate n'a plus qu'à trouver une faille applicative (voir chapitre suivant) sur le logiciel correspondant.

2.2.1.1. Comment s'en protéger ?

Retirer les bannières donnant les versions de logiciel et les messages d'aide ou de bienvenue d'un service réseau en écoute qui peuvent donner des informations sur votre système. Utilisez *netcat* contre vos serveurs pour repérer les services trop « bavards ».

Chapitre 3. Les failles applicatives

Introduction

Nous allons aborder dans ce chapitre les failles liées aux applications utilisées. Nous nous focaliserons principalement sur les failles logicielles : les failles applicatives.

Ces failles peuvent être de natures diverses : problèmes de configuration, problèmes au niveau du code du logiciel, problèmes liés à de mauvaises interprétations de commandes ou de mauvaises exécutions de scripts.

3.1. Les installations par défaut

Lors d'une installation, beaucoup de services peuvent être installés par défaut (un serveur Web, FTP ...). Ces services peuvent contenir les différents types de failles introduites auparavant. L'important est de bien contrôler lors de l'installation, les services qui seront installés sur le système. Pour être bien sûr de soi, il est aussi recommandé de *scanner* la machine pour voir ce qui y tourne. Voir [Section 2.1](#).

Même si certains logiciels ne comportent pas de failles connues, ils peuvent quand même donner des informations aux pirates (voir section [FIXIT]).

3.2. Les mauvaises configurations

Lorsqu'une application est mal paramétrée, elle peut laisser l'accès libre à certaines bases de données sensibles (fichiers de mots de passe, d'utilisateurs) ou de permettre d'exécuter des commandes ou des scripts malveillants.

Il est important de bien lire le manuel avant d'activer un service (RTFM !) et de bien définir «qui fait quoi».

Ce principe est simple : il suffit de bien définir les utilisateurs et les groupes et de limiter leurs droits sur certains types de fichiers et certaines opérations d'exécution de commandes système.

Le plus important est de restreindre au maximum les accès à certains fichiers sensibles et aux commandes systèmes.

3.3. Les bogues

Les bogues sont dus à des erreurs de programmation. Les bogues font apparaître différents types de problèmes de sécurité :

3.3.1. Des dénis de services applicatifs

Ce type de faille empêche le logiciel de fonctionner et ainsi de répondre aux requêtes demandées (d'où l'appellation déni de service). La technique est simple, il suffit d'utiliser un bogue connu qui va faire planter le logiciel assurant un service.

3.3.2. Outrepassement de droits

Les bogues de type dépassement de buffer ou d'exploitation de bogues de format posent de gros problèmes de sécurité. Ils visent majoritairement des applications fonctionnant avec les accès administrateur (*setuid root*) pour permettre à un attaquant d'obtenir un interpréteur de commande au niveau administrateur (*uid root*) sans aucune authentification.

3.3.3. Les scripts

Malheureusement, une mauvaise programmation de scripts ou l'utilisation de fonctions boguées peut être source de failles de sécurité. Il convient d'être très attentif au niveau du développement d'un script.

3.4. Les exploits

Pour exploiter ces bogues, le pirate fait appel à des «exploits». Ces «exploits» sont en fait de petits programmes permettant d'exploiter une faille dans un but précis (obtenir un interpréteur de commandes, accéder à certains fichiers, augmenter ses droits...).

Les exploits peuvent aussi fonctionner à distance, pour l'obtention d'un shell (parfois avec les droits administrateur) sans mot de passe, ni nom d'utilisateur.

3.5. Comment s'en protéger ?

Face aux multiples failles de sécurité des systèmes d'information, seule la *veille* permet de répondre aux objectifs de continuité de service. Pour assurer cette *veille*, les responsables système et réseau disposent de différentes sources d'informations :

Les sites d'informations dédiées sur Internet

Le réseau des *Computer Emergency Response Teams* publie des rapports sur toute nouvelle faille de sécurité. Ces équipes peuvent aussi fournir une assistance en cas de piratage.

A la tête de l'Internet, on trouve le *CERT* (<http://www.cert.org>) de l'université de *Carnegie Mellon*. Au niveau national, on dispose de deux CERTs : le *CERT RENATER* (http://www.renater.fr/Securite/CERT_Renater.htm) dont les *archives des avis de sécurité* (<http://www.cert.uhp-nancy.fr/>) sont publiques et le *Centre d'Expertise gouvernemental de Réponse et de Traitement des Attaques informatiques* (<http://www.certa.ssi.gouv.fr/>).

Sur un plan moins «officiel», les *Archives Bugtraq* (http://citadelle.intrinsec.com/ mailing/current/HTML/ml_bugtraq/) (en anglais) font partie des meilleures sources sur les nouvelles vulnérabilités. Ces archives donnent des descriptions très précises sur des nouvelles failles de sécurité. *Bugtraq France* (<http://www.bugtraq-france.com>) se veut l'équivalent français.

Certains sites comme *.[packet storm security]:.* (<http://packetstormsecurity.nl/>) ou *SecurityFocus* (<http://www.securityfocus.com/>) contiennent aussi de nombreuses informations. Le site *SecurityFocus* (<http://www.securityfocus.com/>) fournit un moteur de recherches thématique pratique pour lister les vulnérabilités liées à un logiciel.

La détection d'intrusion réseau

Les systèmes de détection d'intrusion réseau (*Network Intrusion Detection Systems* ou NIDS) peuvent permettre de repérer les attaques exploitant des failles connues sur certains types de logiciels.

Les correctifs anti-débordement mémoire pour le noyau

Il existe plusieurs outils complémentaires au noyau Linux qui permettent de limiter les possibilités d'exécution d'exploits utilisant les bogues de dépassement de mémoire (pile et/ou tas). *OpenWall* (<http://www.openwall.com/>) et *grsecurity* (<http://www.grsecurity.org/>) sont deux exemples caractéristiques.

La limitation du nombre de programmes s'exécutant avec les droits administrateur

Il est toujours bon de repérer les programmes s'exécutant avec les droits administrateur. Ainsi, vous pouvez changer leurs droits pour qu'ils ne deviennent pas un point critique pour

la vulnérabilité du système. Sous linux, la simple commande : **# find / -perm +6000** vous permettra de lister tous les programmes s'exécutant avec les droits administrateur.

Chapitre 4. Les outils indispensables pour la protection

Introduction

Les pare-feux (*firewalls*), les tunnels et les systèmes de détection d'intrusion aux niveaux hôte et/ou réseau (IDS : *Intrusion Detection System*/NDIS : *Network Intrusion Detection System*) sont des outils indispensables pour détecter, parer ou éviter de nombreuses attaques. Nous les décrirons dans ce chapitre, ainsi que la manière de les utiliser de façon optimale.

4.1. Le pare-feu *firewall*

La configuration d'un pare-feu peut s'avérer être un sujet très difficile à traiter. Cette configuration est surtout établie en fonction de vos besoins personnels.

L'objectif de ce chapitre est de donner des conseils à suivre pour bien utiliser un pare-feu. Ensuite, nous nous intéresserons aux différentes méthodes d'attaques contre les pare-feux.

4.1.1. La configuration

Pour bien configurer son pare-feu, il suffit de bien respecter les conseils suivants :

- Essayez de limiter l'accès à votre réseau à des utilisateurs connus utilisant une adresse IP statique. Vous pourrez ainsi rejeter toutes les autres requêtes venant d'utilisateurs utilisant une adresse IP non autorisée. Vous effectuez de la sorte un filtrage au niveau IP.
- Fermez tous les ports en écoute sur les différents serveurs et ouvrez seulement ceux dont vous avez besoin.
- Filtrez ces ports, c'est à dire rejetez toutes les autres requêtes sur les autres ports que ceux en écoute.
- Empêchez toutes les connexions sortantes sur des services non autorisés. Pour cela, il suffit de définir un nombre limité de services auxquels les serveurs et les clients peuvent accéder (mail, ftp, web...). Ensuite, il faut configurer le firewall pour rejeter les connexions depuis l'intérieur vers l'extérieur sur des services différent de ceux définis.

4.1.2. Les attaques contre les firewalls

La première étape lors d'une attaque est de déterminer les règles actives sur le pare-feu. Le but est simple : savoir quels ports ne sont pas filtrés.

4.1.2.1. Avec le *scanner*

Nmap peut déterminer quels ports sont ou ne sont pas filtrés. Voici un exemple :

```
[root@nowhere.net /root]# nmap 192.168.1.2

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.1.2) :
(The 1549 ports scanned but not shown below are in state : closed)
Port State Service
21/tcp filtered ftp
22/tcp filtered ssh
111/tcp open sunrpc
515/tcp open printer
1024/tcp open kdm

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Les ports 21 (ftp) et 22 (ssh) sont filtrés.

4.1.2.2. Comment s'en protéger ?

Protégez-vous contre le scanning (voir section FIXIT scan). Si le scan ne marche pas, d'autres méthodes peuvent être utilisées comme le *firewalking*.

4.1.2.3. Le firewalking

Cette technique de dévoilement des règles de firewalling repose sur un unique champ de l'en-tête IP, le TTL (*Time To Live*), ce champ représentant la durée de vie d'un paquet. Il est fixé dès son envoi par la pile de protocole du système et est diminué d'une unité à chaque fois qu'il traverse un équipement assurant le routage ; quand ce champ est égal à 0, le paquet est jeté à la poubelle.

Chaque passage d'un routeur à un autre est appelé un saut.

C'est ce champ qui est utilisé par le logiciel `traceroute`. Pour tracer une route, le logiciel envoie un premier paquet UDP avec un TTL de 1 ; au premier routeur, le TTL est décrémenté à 0. Le routeur renvoie un message ICMP de type 11 ICMP `TTL Exceeded`, ce qui permet de récupérer l'adresse du premier routeur. Ensuite `traceroute` va envoyer un deuxième paquet avec un TTL de 2, il sera décrémenté au passage du premier routeur (TTL=1). Le deuxième routeur va le recevoir et le décrémenter : le champ TTL sera de nouveau égal à 0. Le deuxième routeur renverra donc un message d'erreur : on récupère ainsi l'adresse du deuxième routeur.

Si il y a N sauts jusqu'au réseau final, on réitère l'opération N fois.

On peut utiliser cette technique avec différents protocoles : UDP, TCP, ICMP.

Le *firewalking* repose sur cette technique pour déterminer les règles actives sur un pare-feu. *firewalk* (<http://www.packetfactory.net/firewalk/>), le programme implémentant le *firewalking*, détermine le nombre de routeurs entre la machine source et la machine cible (située derrière le firewall). Ensuite, il envoie des paquets tests avec un TTL égal à ce nombre de routeurs + 1. Si le paquet est accepté, il traverse le firewall et on obtient une réponse ; sinon il n'y a aucune réponse.

firewalk (<http://www.packetfactory.net/firewalk/>) envoie différents types de paquets (TCP, UDP) pour déterminer les règles de firewalling. Néanmoins, de nombreux paramètres comme la congestion du réseau, le nombre de routeurs séparant la cible et l'émetteur peuvent fausser l'analyse.

4.2. Les systèmes de détection d'intrusion (HIDS/NIDS)

Système de détection d'intrusion

Intrusion Detection System

IDS

Les outils les plus pratiques ! Ces utilitaires permettent de détecter une attaque et de vous en informer. Un IDS analyse tout ce qui se passe sur une station. Il détecte les outrepassements de droits (obtention du compte root d'une manière suspecte) et d'autres types d'attaques, il contient une base de données sur différentes vulnérabilités.

Système de détection d'intrusion réseau

Network Intrusion Detection System

NIDS

Un NIDS travaille de la même manière, mais sur les données transitant sur le réseau. Il peut détecter en temps réel une attaque s'effectuant sur l'une des vos machines. Il contient une base de données avec tous les codes malicieux et peut détecter leurs envois sur une des machines. Le NIDS travaille comme un *sniffer* (voir section FIXIT sniffer), sauf qu'il analyse automatiquement les flux de données pour détecter une attaque.

Cette section présentera deux systèmes de détection d'intrusion : un NIDS appelé *Snort* et IDS hybride *Prelude*. Il est à noter que ces outils sont distribués comme logiciel libre. Je ne rappellerai pas que le logiciel libre a une avance considérable dans le domaine de la sécurité par rapport à ses concurrents «propriétaires».

4.2.1. Prelude-NIDS

Prelude Hybrid IDS (<http://www.prelude-ids.org/>) est un des détecteurs d'intrusions les plus connus. Prelude est disponible et libre sur les plateformes Linux, FreeBSD et Windows.

Prelude possède une architecture modulaire et distribuée. Modulaire, car ses composants sont indépendants, et peuvent être facilement mis à jour. Distribuée, car ces composants indépendants interagissent les uns avec les autres. Cela permet d'avoir divers composants installés sur différentes machines et de réduire ainsi la surcharge d'applications.

Ces différents composants sont les sondes et les managers. Les sondes peuvent être de deux types : réseau ou local. Une sonde réseau analyse tout le trafic, pour y détecter d'éventuelles signatures d'attaques. La sonde locale assure la surveillance d'une seule machine, il analyse le comportement du système pour y détecter des tentatives d'exploitation de vulnérabilités internes. Les sondes signalent les tentatives d'attaques par des alertes. Ces alertes sont reçus par le manager qui les interprète et les stocke.

Pour une description complète de Prelude (installation, configuration et utilisation) consultez ce document : <http://lehmann.free.fr/PreludeInstall/InstallPrelude.html>

4.2.2. Snort

Snort téléchargeable librement sur www.snort.org est un NIDS lui aussi. Il n'est pas structuré comme Prelude. Snort est un programme "monolithique", il ne comporte pas de module comme Prelude, ce qui peut rendre son implémentation dans un réseau un peu moins souple que Prelude. Snort fonctionne en trois modes (Sniffer, PacketLogger et NIDS). Les deux premiers modes ne sont pas intéressants pour la détection d'intrusion. Le troisième mode permet lui d'analyser le trafic réseau pour y détecter d'éventuelles attaques.

Pour une description complète de Snort (installation, configuration et utilisation) consultez ce site : <http://www.snort.org/docs/> (en anglais).

4.3. Le tunneling

Nous allons décrire dans cette section différentes méthodes pour sécuriser vos transactions, c'est-à-dire créer un VPN (Virtual Private Network). Un réseau privé virtuel (VPN) est utilisé pour établir des communications sécurisées en s'appuyant sur un réseau existant non sécurisé. Le principal outil utilisé pour la création de VPN est IPsec.

IPsec est facultatif sur IPv4 mais est obligatoire sur IPv6. IPsec a d'autres avantages que la sécurisation du trafic, il permet par exemple d'économiser la bande passante grâce à la compression des en-têtes des paquets.

IPsec fonctionne sous deux modes différents : le mode transport et le mode tunnel. Ces deux modes seront décrits dans ce qui suit.

IPsec est composé de plusieurs protocoles différents : AH, ESP, IPcomp et IKE.

4.3.1. Le protocole AH

Le protocole AH (Authentication Header) permet de garantir l'authenticité des paquets échangés en leur inscrivant une somme de contrôle (de l'en-tête IP jusqu'à la fin du paquet) chiffrée.

4.3.2. Le protocole ESP

Le protocole ESP (Encapsulating Security Payload) crypte toutes les données du paquet garantissant leur confidentialité.

4.3.3. Le protocole IPcomp

Le protocole IPcomp (IP payload compression) permet de compresser un paquet avant de le chiffrer avec ESP.

4.3.4. Le protocole IKE

Le protocole IKE (Internet Key Exchange) est utilisé pour l'échange des clés utilisés pour l'encryptage.

4.3.5. Les deux modes de fonctionnements de IPsec

AH, ESP et IPcomp fonctionnent dans le mode transport ou le mode tunnel. Le mode "transport" crypte directement les échanges entre deux machines. Le mode "tunnel" encapsule les paquets encryptés dans de nouveaux en-tête IPv4/IPv6. Il est conçu pour les passerelles VPN.

4.3.6. Les limitations d'IPsec

IPsec pose quelques problèmes dus à son implémentation. Certains problèmes apparaissent au niveau des messages de broadcast et multicast. IPsec est difficile à filtrer sur les firewalls existants. Il est aussi impossible à gérer pour les translations d'adresse (NAT).

4.3.7. Documents

Description générale des tunnels et implémentation sous Linux :

<http://www.miscmag.com/articles/index.php3?page=315>

<http://www.securiteinfo.com/crypto/IPSec.shtml> Description générale des tunnels et

implémentation sous Windows : <http://www.laboratoire-microsoft.org/articles/network/ipsec/>

4.4. Nessus

Nessus est un outil de sécurité permettant de scanner une ou plusieurs machines. Il permet aussi de tester différentes attaques pour savoir si une ou plusieurs machines sont vulnérables.

Il est très utile lors de tests de pénétration (pen test) et fait gagner un temps incroyable.

Nessus se compose d'une partie serveur (qui contient une base de données regroupant différents types de vulnérabilités) et une partie client. L'utilisateur se connecte sur le serveur grâce au client et après authentification, il ordonne au serveur de procéder aux tests d'une ou plusieurs machines. Le client reçoit ensuite les résultats du test.

Nessus est disponible sous Linux et Windows, et il est entièrement gratuit.

4.4.1. Pour obtenir tout sur Nessus

Pour télécharger les sources, binaires, ou différents documents concernant Nessus, consultez le site : <http://www.nessus.org>

4.5. User Mode Linux - UML

User Mode Linux est un dispositif permettant de lancer un ou plusieurs noyau(x) linux dans l'espace utilisateur (comme un simple programme). En clair, il permet d'avoir plusieurs machines virtuelles sur une seule machine physique hôte exécutant Linux. Les avantages sont nombreux :

- Si une machine virtuelle plante, le système hôte n'est pas affecté.
- Un utilisateur sera root sur une machine virtuelle, mais pas sur le système hôte.
- Il permet aussi de tester différents paramètres noyaux sans se soucier des conséquences.

User Mode Linux permet notamment de s'affranchir de chroot (pour, par exemple, la réalisation de serveurs FTP) et de toutes ses failles de sécurité.

4.5.1. Documents

La page web de User Mode Linux : <http://user-mode-linux.sourceforge.net> Un autre lien intéressant pour User Mode Linux et la sécurité : http://www.rstack.org/oudot/20022003/7/7_rapport.pdf

Chapitre 5. Surveillance - Dissimulation - Maintien d'accès

Introduction

Nous présenterons dans ce chapitre les programmes utilisés par les pirates pour dissimuler, surveiller et maintenir leur accès sur un système d'information. Nous présenterons les moyens de s'en protéger.

5.1. Les chevaux de Troie

Le principe du «Cheval de Troie» est facile à comprendre. Un programme ou un code malveillant est intégré à une application par ajout ou par modification de son code. Ainsi lors de l'exécution de ce programme inoffensif, le bout de code malveillant pourra exécuter des commandes spécifiques (récupération de fichiers de mot de passe, altération du système, etc.) à l'insu de l'utilisateur.

5.1.1. Comment s'en protéger ?

La plupart des antivirus peuvent détecter les chevaux de Troie. Néanmoins, comparer la signature numérique accompagnant les fichiers (cela se fait par un calcul reposant sur un algorithme de chiffrement appliqué à l'ensemble du fichier) avec la sienne permet de savoir directement si l'on est infecté.

Il est aussi conseillé de consulter les sites suivants pour vérifier que vos programmes ne contiennent pas de chevaux de Troie :

- Le *CERT* (<http://www.cert.org>) (*Computer Emergency Response Team*) est un organisme s'occupant des problèmes de sécurité sur Internet. Il recense les différents problèmes de sécurité et publie des articles (*advisories*) pour les décrire.
- Bugtraq : http://citadelle.intrinsec.com/ mailing/current/HTML/ml_bugtraq/

5.2. Les backdoors

Les backdoors sont des accès cachés sur un système ou sur une application. Le principe d'une backdoor est similaire à celui du cheval de Troie. L'objectif est de modifier ou d'utiliser un programme pour accéder discrètement à un ordinateur distant, modifier le comportement d'un programme, devenir administrateur.

5.2.1. Les backdoors présentes dans les logiciels.

Parfois, certains logiciels (messagerie, utilitaires systèmes) peuvent contenir des backdoors, c'est-à-dire que, pour certaines commandes suivies d'arguments particuliers ou avec un mot de passe bien défini, le logiciel peut avoir un comportement différent (permettre à l'utilisateur de devenir root, renvoyer un shell système à l'utilisateur, etc.).

Ces "trappes" sont incluses directement dans le code du logiciel. Certains développeurs sont soucieux de posséder un accès sur tous les systèmes utilisant leurs logiciels. Par exemple, Ken Thompson, l'un des pères d'UNIX, avoue avoir modifié l'application `/bin/login` en permettant l'accès direct au système par la saisie d'un mot de passe précompilé en dur. Thompson pouvait ainsi visiter tous les systèmes utilisant son application modifiée. Parfois, certains pirates diffusent des applications infestées de backdoors.

5.2.1.1. Comment s'en protéger ?

Il convient de télécharger ses applications sur le site du distributeur ou du programmeur. Utiliser des serveurs de téléchargement non liés à l'auteur de l'application peut se révéler dangereux.

Il est aussi recommandé de vérifier les checksums s'il sont donnés par le développeur.

Il est aussi bon de consulter des listes de diffusion comme bugtraq pour savoir si la version de logiciel que vous utilisez ne comporte pas de backdoors.

5.2.2. Les backdoors dédiées aux connexions à distance

Ces backdoors peuvent très bien faire partie de la première catégorie. Comme il l'a été montré, certains logiciels peuvent autoriser un accès pour un mot de passe particulier. Toutefois, ce paragraphe va se focaliser sur des applications en écoute sur un port bien défini utilisées par les pirates pour obtenir un shell. Un logiciel préalablement installé par le pirate est en attente de connexion sur un port discret. La plupart de ces programmes sont en écoute sur des numéros de ports ayant une valeur assez élevée (supérieur à 5000). Le pirate n'a plus qu'à se connecter sur ce programme pour récupérer son accès sur la machine.

5.2.2.1. Comment s'en protéger ?

Nmap peut se révéler être une aide précieuse pour les débusquer. Si, en procédant au scan d'une machine, vous constatez qu'un port non autorisé est en écoute, il serait bon de vérifier celui-ci.

Les sites à consulter :

- Le CERT (<http://www.cert.org>) en anglais.
- Bugtraq : http://citadelle.intrinssec.com/ mailing/current/HTML/ml_bugtraq/ (en anglais) et <http://www.bugtraq-france.com>.

5.3. Les Rootkits

Le rootkit est un programme permettant d'automatiser la dissimulation et l'effacement des traces d'un pirate sur une machine. L'objectif d'un rootkit est de modifier les commandes permettant d'administrer le système, de cacher les ports ouverts par le pirate.

Les premiers rootkits étaient assez basiques, ils modifiaient juste les commandes ls, ps, netstat.

L'administrateur pouvait détecter ces modifications sur les logiciels concernés. Alors une seconde génération de rootkits apparut. Il faut savoir que des commandes comme ps, ls ... font appels à des bibliothèques partagées pour fonctionner. Les nouveaux rootkits modifiaient donc le code de ces bibliothèques pour modifier le comportement de ces commandes à l'avantage du pirate.

Encore une fois, ceci était détectable. Donc une troisième génération de rootkits est née afin de modifier directement le comportement du noyau, par le biais de modules chargés en mémoire (LKM). C'est à l'heure actuelle la dernière génération.

Différents rootkits sont disponibles sur Linux.

Je ne donnerai (volontairement) pas dans cette partie une description complète de l'utilisation des rootkits. Cela n'a aucun intérêt pour ce guide.

La plupart des rootkits utilisent le principe des backdoors ([Section 5.2](#)) pour permettre au pirate de se connecter selon son envie sur un système.

5.3.1. Comment s'en protéger ?

1. Les checksums. Une base de données de checksums sur les différents fichiers système peut déjà constituer une bonne parade. Je vous conseille d'effectuer des checksums à la fin d'une installation sur les différents fichiers comme ls, ps, stat ifconfig, etc. et sur les différentes bibliothèques partagées.

Cette base de donnée devrait être stockée sur un CDROM ou tout autre support non réinscriptible.

2. Compiler les programmes vitaux en statique. Comme je l'ai dit précédemment, certaines commandes font appels à des bibliothèques partagées et des utilitaires comme "md5sum" (qui sert à faire des checksums) sous Linux font appels à des bibliothèques partagées. D'où son comportement pourrait être modifié indirectement par un rootkit attaquant les bibliothèques partagées. Pour éviter ce genre de désagrément, compilez une partie des programmes vitaux en statique, ainsi vous disposerez d'une trousse de secours en cas d'infection par rootkits.

Bien sûr, pour compiler les programmes vitaux en statique, faut-il encore disposer d'un OS qui permette d'accéder aux sources de ces programmes vitaux...

3. Chkrootkit. Chkrootkit (pour CHeCK ROOTKIT) vous permet de détecter la présence d'un rootkit, il fonctionne sous Linux (FreeBsd...) et est téléchargeable librement sur www.chkrootkit.org.
4. Compilez votre noyau en statique. Vous éviterez ainsi le chargement de modules externes.

5.4. L'interception des mots de passe en réseau.

Une autre technique utilisée pour collecter des informations (mots de passe par exemple) est l'utilisation d'un sniffer. Le sniffer place la carte réseau dans le mode transparent (promiscuous), ce qui veut dire que la carte intercepte tous les paquets sur le segment réseau, même ceux qui ne lui sont pas destinés.

Plusieurs types de sniffers existent ; certains affichent les données interceptées brutes comme Tcpdump disponible sur www.tcpcdump.org, ce qui donne lieu à des fichiers de log très volumineux. D'autres sniffers permettent de récupérer les mots de passe en les affichant directement à l'écran associé avec le login, l'adresse du client et celle du serveur (comme dsniiff disponible sur www.packetstormsecurity.org).

Ethereald disponible sur www.packetstormsecurity.org permet par exemple d'afficher toutes les transactions ayant cours sur le réseau.

Cependant, le sniffer reste un outil puissant pour la détection d'intrusion car, premièrement, il garde une trace de tous les échanges ayant cours sur le réseau. Deuxièmement, il se révèle très utile pour démasquer un scan (un grand nombre de paquets envoyés d'un seul coup), de tracer l'adresse d'un pirate, de voir si des commandes particulières sont demandées sur le réseau.

La plupart des rootkits contiennent un programme pour sniffer.

Les NDIS utilisent un sniffer pour analyser les transactions réseau.

5.4.1. Comment s'en protéger ?

Là, c'est très difficile. Un sniffer est passif, il n'envoie aucun paquet, il ne fait qu'intercepter. Mais la carte réseau étant en mode transparent, son comportement s'en trouve changé, son temps et sa façon de répondre à certains paquets sont modifiés. On peut détecter la présence d'un sniffer grâce à ce changement de comportement. Le programme AntiSniff disponible sur windows NT et Linux à l'adresse <http://packetstormsecurity.nl/sniffers/antisniff/> de Lopht Heavy Industries

peut envoyer des paquets "tests" et en déduire si la carte est en mode transparent donc susceptible de sniffer.

Une deuxième parade pour déjouer le sniffing est de "tunneler" toutes les transactions. Cela veut dire encrypter toutes les transactions réseaux. Utiliser IpvSec ou des VPN, ssh sur votre réseau s'avère être une défense efficace contre le sniffing.

L'utilisation de tunnels est traitée dans la section #x1-460004.3.

Chapitre 6. Dispositifs destructeurs

Introduction

Les dispositifs destructeurs sont utilisés pour paralyser, saturer ou détruire un système d'information. Ils constituent l'espèce la plus nuisible dans le domaine de la sécurité car ils peuvent être la source de perte de données. Le but de ce chapitre est d'expliquer leurs fonctionnements et la façon de les combattre.

6.1. Le virus

Le virus est un programme dont le seul but est de consommer ou de paralyser des ressources système. Le virus s'autoduplique pour mieux infecter le système, il se propage en infectant tour à tour les fichiers. Les effets d'une contamination varient : fichiers effacés, disque dur formaté, saturation des disques, modification du MBR, etc.

La grande majorité d'entre eux existent sur les plates-formes Microsoft, ils infectent en particulier les fichiers COM ou EXE. De plus, de nouvelles formes sont apparues comme les macro-virus qui attaquent les fichiers de données (word ou excel).

Les systèmes UNIX ne sont pas épargnés ! Les administrateurs UNIX doivent faire face à des virus comme Winux. Néanmoins, la gestion des droits sous UNIX se révèle être un facteur limitant pour la propagation de virus.

Les virus sont de plus en plus évolués, ils peuvent s'automodifier pour échapper à une éventuelle détection (virus polymorphes). D'autres types peuvent tenter de leurrer le système en s'installant dans des secteurs défectueux ou non utilisés (virus furtifs) ...

6.1.1. Comment s'en protéger ?

Les anti-virus commerciaux comme Norton Antivirus ou McAfee VirusScan sont de bons outils pour traquer les virus. Toutefois, il convient de les mettre régulièrement à jour pour profiter pleinement de leurs capacités.

Il est aussi important de suivre l'évolution et l'apparition de nouveaux virus ; pour cela, consulter les sites (ainsi que pour tous les autres dispositifs destructeurs décrits dans ce chapitre) :

- Le CERT (<http://www.cert.org>) en anglais.
- Les sites du CNRS : <http://www.services.cnrs.fr/wws/info/sos-virus> et <http://www.cnrs.fr/Infosecu/Virus.html>.

6.2. Les vers

Les vers sont du même acabit que les virus, sauf qu'ils n'utilisent pas nécessairement un fichier pour se propager. Ils sont aussi capables de se dupliquer et de se déplacer au travers d'un réseau informatique. Les vers utilisent différents supports pour se propager.

Les vers simples utiliseront des failles propres à certains logiciels (exemple du ver de Morris en 1988 qui paralysa une grande partie de l'Internet).

Les macro-vers utiliseront les pièces jointes contenant des documents bureautiques infectés (exemple du ver Nimda).

Les vers d'email sont contenus dans une pièce jointe comprenant un code malicieux exécuté automatiquement par le logiciel de courrier électronique ou manuellement par l'utilisateur.

6.2.1. Comment s'en protéger ?

Comme pour les virus, l'antivirus se révèle être une parade efficace. Consultez les listes citées dans la section #x1-730006.1.

6.3. Les bombes logiques

Les bombes logiques sont aussi néfastes que les virus ou les vers et sont la cause de dégâts similaires. La différence est que la bombe logique a besoin d'un détonateur pour s'activer, c'est-à-dire qu'elle attend une date ou une action bien précise de l'utilisateur pour exploser.

6.3.1. Comment s'en protéger ?

Utilisez un anti-virus performant (Mc Afee, Norton ...) régulièrement mis à jour.

Consultez les sites décrits dans la section #x1-730006.1.

6.4. Les attaques par déni de services

Ce type d'attaque est la plus énervante qui soit. Elles ont pour but de saturer le réseau ou le système.

6.4.1. Le SYN flood

Cette technique consiste à saturer un serveur en envoyant une multitude de paquets TCP avec le flag SYN armé, cela aura pour but de créer une multitude de connexions demandant un grand nombre de ressources système.

La plupart des attaques par SYN-flood sont bien détectées par différents firewalls.

6.4.1.1. Comment s'en protéger ?

Exemple avec **iptables** limitant les demandes d'établissement de connexion TCP acceptées à une par seconde:

```
[root@nowhere /root]# iptables -A FORWARD -p tcp --syn -m limit --limit 1/second -J ACCEPT
```

Pour plus de détails sur cette commande, je vous conseille de lire *Utiliser iptables :Spécifications de filtrage* (<http://www.linux-france.org/prj/inetdoc/guides/packet-filtering-HOWTO/packet-filtering-HOWTO-7.html#ss7.3>).

6.4.2. L'UDP Flood

De la même manière que pour le SYN flooding, l'attaquant envoie un grand nombre de requêtes UDP sur une machine. Le trafic UDP étant prioritaire sur le trafic TCP, ce type d'attaque peut vite troubler et saturer le trafic transitant sur le réseau.

La plus célèbre attaque utilisant l'UDP-flooding est le *Chargen Denial of Service Attack*. Un pirate envoie une requête sur le port echo d'une machine A indiquant comme port source celui du port chargen d'une machine B. Le service chargen de la machine B renvoie un caractère sur le port echo de la machine A.

Ensuite le service echo de A renvoie ce caractère sur chargen. chargen le reçoit, en ajoute un autre et les renvoie sur le port echo de A qui les renvoient à son tour sur chargen ... et cela continue jusqu'à la saturation de la bande passante.

6.4.2.1. Comment s'en protéger ?

Il est conseillé de désactiver les services chargen et echo.

Si vous ne voulez pas désactiver chargen et echo, configurez votre firewall pour éviter le *Chargen Denial of Service Attack* en limitant le trafic UDP. Exemple avec **iptables** :

```
[root@nowhere /root]# iptables -A FORWARD -p udp -m limit --limit 1/second -J ACCEPT
```

6.4.3. La fragmentation de paquets

Plus connu sous le nom de *Teardrop Attack*, *Bonk* ou encore *Boink*, cette attaque utilise une faille propre à certaines piles TCP/IP. Cette vulnérabilité concerne la gestion de la fragmentation IP.

Ce problème apparaît lorsque la pile reçoit le deuxième fragment d'un paquet TCP contenant comme donnée le premier fragment. La pile TCP/IP peut s'avérer incapable de gérer cette exception et le reste du trafic.

Cette faille est très connue sur les piles de Windows 95 et 98.

6.4.4. Ping of death

Le principe est d'envoyer un paquet ICMP avec une quantité de données supérieure à la taille maximale d'un paquet IP . Encore une fois, la pile peut s'avérer incapable de gérer cette exception et le reste du trafic.

6.4.5. Attaque par réflexion : Smurfing

Cette attaque est basée sur le protocole ICMP. Lorsqu'on envoie un ping à un réseau en broadcast (par exemple 255.255.255.0), le paquet est envoyé à chacune des machines du réseau.

Un pirate envoie un ping en broadcast sur un réseau (A) avec une adresse IP source correspondant à celle de la machine cible (B). Le flux entre le port ping de la cible (B) et du réseau (A) sera multiplié par le nombre de machines sur le réseau (A).

Cela conduit à une saturation de la bande passante du réseau (A) et du système de traitement de paquets de (B).

6.4.5.1. Comment s'en protéger ?

Configurez votre firewall pour limiter le trafic ICMP. Exemple avec **iptables** :

```
[root@nowhere /root]# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/second
```

Pour plus de détails sur cette commande, je vous conseille de lire *Utiliser iptables :Spécifications de filtrage* (<http://www.linux-france.org/prj/inetdoc/guides/packet-filtering-HOWTO/packet-filtering-HOWTO-7.html#ss7.3>).

6.4.6. Dénis de services distribués

Plusieurs types d'attaques sont lancées en parallèle à partir de plusieurs sources.

6.4.7. Bombes e-mail

Le *mail bombing* consiste à envoyer de gros ou de nombreux fichiers à un utilisateur pour saturer sa boîte de réception de courrier électronique.

6.4.7.1. Comment s'en protéger ?

La plupart des logiciels de contrôle de contenu assure un filtrage du courrier pour détecter ce type d'attaque

Chapitre 7. Sécurisation des mots de passe

Introduction

Le but de ce chapitre est de donner au lecteur toutes les informations nécessaires sur les techniques utilisées pour tester la résistance des protections par mot de passe.

Il faut savoir que les mots de passe utilisés sur un système d'information sont encryptés pour garantir leur confidentialité. Ces mots de passe encryptés sont stockés dans des listes de mots de passe sur des fichiers systèmes prédéfinis.

Un pirate peut fort bien récupérer ces listes et tester la fiabilité des mots de passe. Il utilise pour cela l'outil adéquat : un perceur de mot de passe.

La plupart des algorithmes d'encryptage repose sur l'utilisation de fonctions à sens unique. Ceci veut simplement dire qu'il est impossible de décrypter le mot de passe à partir sa forme encryptée. L'attaque consiste alors à encrypter différentes combinaisons de caractères et de comparer cette forme encryptée à celle du mot de passe voulu. Si les deux chaînes correspondent, alors la suite de caractères est celle du mot de passe.

Il y a deux types d'attaques pour le craquage de mots de passe qui seront définies dans ce chapitre.

7.1. L'attaque par dictionnaire

Le programme utilise une liste de mots prédéfinis dans un fichier externe. Cette liste est appelée un dictionnaire ; ces mots sont la plupart du temps ceux provenant d'un dictionnaire contenant les mots du langage courant. Le programme les encrypte avec l'algorithme d'encryptage adéquat un par un et les compare au mot de passe encrypté.

Ce type d'attaque est très rapide. Un mot de passe mal choisi est vite découvert.

7.2. Le brute forcing

Si l'attaque par dictionnaire ne marche pas, le programme peut générer des mots de passe avec une suite aléatoire de caractères, les encrypter et les comparer au mot de passe à découvrir. Avec un mot de passe suffisamment long (supérieur à 8 caractères), cette méthode a peu de chance d'aboutir. Si, de plus, des caractères spéciaux sont ajoutés comme des signes de ponctuation, la méthode peut se révéler inefficace.

Il existe différents logiciels de perçage de mots de passe en fonction du type d'encryptage (DES, MD5, special Microsoft ...).

7.3. Tester la fiabilité de vos mots de passe !

Sous UNIX

Sous UNIX, la liste des mots de passe des utilisateurs système est divisée en deux fichiers `/etc/shadow` et `/etc/passwd` ou réunis seulement dans le fichier `/etc/passwd`. Le type d'encryptage peut être du MD5, DES, l'algorithme DES a été adopté par la NSA comme standard à la fin des années 70. Le DES est distribué publiquement mais il a été développé dans le secret. Certains suspectent le gouvernement américain de s'être réservé une "gâche secrète" pour une décryptage plus rapide, je vous conseille d'utiliser d'autres algorithmes à la place., RSA ...

Pour tester la résistance de vos mots de passe, le logiciel *John The Ripper* John The Ripper est disponible sur <http://www.openwall.com/john/> peut s'avérer être une bonne aide. Il

supporte un grand nombre d'algorithmes d'encryptage, présente un important paramétrage des attaques. John the Ripper est un programme distribué librement.

Sous Windows

Pour tester la fiabilité des mots sous Windows, l'administrateur pourra utiliser le logiciel John the RipperIdem. sur Windows ou sur Unix ou le logiciel LophtCrack de Lopht Heavy Industries Lophtcrack qui, lui, n'est pas distribué gratuitement (enfin pas pour les versions récentes).

Sous Windows 9x, les mots de passe sont dispersés dans le répertoire racine de windows dans différents fichiers d'extension ".PWL" portant comme nom celui de l'utilisateur.

Le chiffrement utilisé pour générer les mots de passe PWL est très faible. Le programme Cain permet de tester leur fiabilité.

7.4. Choisir le bon mot de passe

N'utilisez pas des mots de votre langage courant ou des suites de chiffres !

Choisissez des mots de passe longs, avec une suite de caractères totalement aléatoires et avec des caractères spéciaux, alternez les majuscules et les minuscules.

Choisissez une phrase et alternez les majuscules et minuscules avec les premières lettres de chaque mot en tenant compte de la ponctuation. Par exemple :

A demain, Je t'Aime mon Amour.

donne : Ad,Jt'AmA, qui est un mot de passe assez costaud.

7.5. Prévenir l'utilisateur

N'hésitez pas à organiser des réunions, faire circuler différents documents pour informer vos utilisateurs des problèmes de fiabilité des mots de passe.

Chapitre 8. La base des attaques réseaux

Introduction

Dans ce chapitre, nous décrirons les principes sur lesquels reposent de nombreuses attaques réseaux (notamment celles décrites dans le chapitre 9), ainsi que les règles à respecter pour les éviter ou les parer.

8.1. Détournement de flux

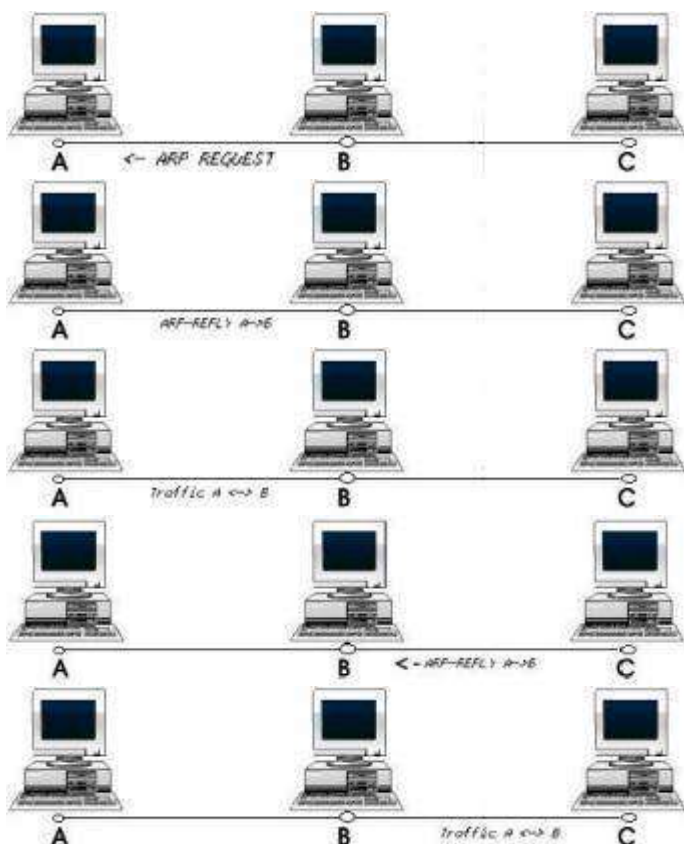
Les techniques de détournement de flux servent à rediriger le flux réseau vers un client, vers un serveur, ou vers une autre machine.

8.1.1. ARP-Poisoning

Toute carte réseau possède une adresse physique. C'est cette adresse qui lui permet de recevoir les paquets qui lui sont destinés sur le réseau local. Cette adresse physique est associée à l'adresse IP grâce au protocole ARP. La table de correspondance entre les adresses IP et les adresses physiques est contenue dans le cache ARP. Lorsqu'un échange doit s'établir entre 2 machines du réseau local, ces deux machines envoient des requêtes ARP avec l'adresse IP du récepteur, associée à un champ vide pour son adresse physique. Ce récepteur va renvoyer son adresse physique dans une réponse ARP.

Si un attaquant envoie un message de réponse ARP avec son adresse physique correspondant à l'adresse IP du récepteur, tout le flux IP dirigé vers le récepteur sera redirigé vers l'attaquant. On dit qu'il a empoisonné le cache ARP du récepteur.

Illustration :



8.1.1.1. Comment s'en protéger ?

La solution la plus immédiate consiste à saisir manuellement sur chaque poste la table de toutes les adresses physiques présentes sur le réseau local. Si elle est immédiate, cette solution est quasiment inapplicable compte tenu du nombre d'hôtes connectés au réseau local.

Une solution correcte consiste à mettre en place un serveur DHCP avec une liste «fermée» de correspondance entre adresses physiques (MAC) et IP. Relativement à la solution précédente, la liste exhaustive des adresses physiques est centralisée sur le serveur DHCP. On peut ensuite configurer la journalisation du service pour que toute requête DHCP relative à une adresse MAC inconnue génère un courrier vers l'administrateur système.

Enfin, On peut utiliser sous UNIX, un logiciel spécialisé : `arpwatch` (<ftp://ftp.ee.lbl.gov/>) qui permet de surveiller tout le trafic ARP.

Les NIDS peuvent aussi détecter ce type d'attaques (notamment Prelude-IDS).

8.1.1.2. Documents

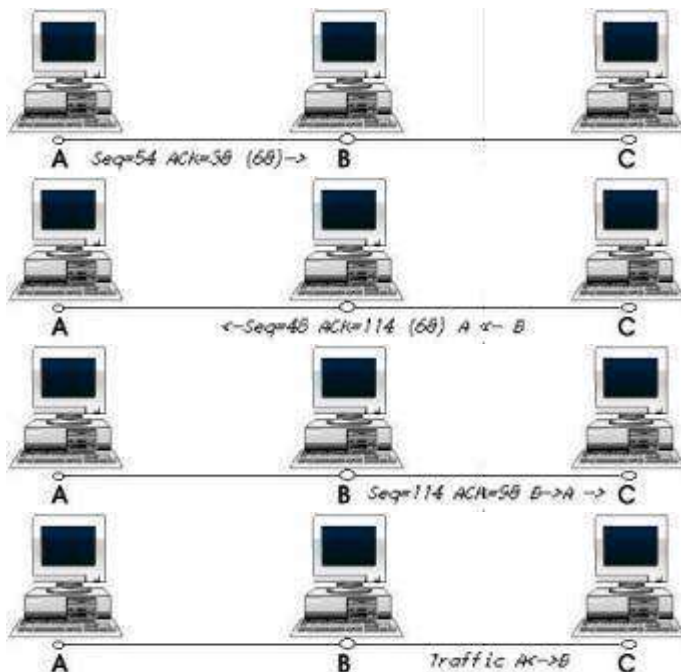
L'article *Le arp-poisoning* (<http://www.securite.teamlog.com/publication/6/10/102/>) est une bonne introduction à la problématique.

La section *Address Resolution Protocol (ARP)* (<http://linux-ip.net/html/ether-arp.html>) du *Guide to IP Layer Network Administration with Linux* (<http://linux-ip.net/html/index.html>) montre toutes les possibilités d'interaction sur le protocole ARP avec le noyau Linux.

8.1.2. Désynchronisation TCP

L'ARP-Poisoning permet de rediriger tout le trafic IP mais, si l'attaquant n'a besoin que du trafic TCP, il peut interférer entre une connexion client-serveur pour rediriger le flux du client vers lui. La synchronisation TCP est assurée par les numéros de séquences TCP. Si, pendant un échange, l'attaquant envoie des paquets malformés au client avec une adresse IP correspondant à celle du serveur en y plaçant des mauvais numéros de séquences, le client va croire qu'il a perdu la connexion et stoppera ses échanges avec le serveur. Mais si l'attaquant envoie les bons numéros de séquences au serveur, il récupérera la connexion pour lui.

Illustration :



8.2. Man In the Middle - MITM

Les attaques de type *Man-In-the-Middle* sont très faciles à comprendre. *Man-in-the-Middle* signifie l'homme du milieu. Cette attaque fait intervenir trois protagonistes : le client, le serveur et l'attaquant. Le but de l'attaquant est de se faire passer pour le client auprès du serveur et se faire passer pour le serveur auprès du client. Il devient ainsi l'homme du milieu. Cela permet de surveiller tout le trafic réseau entre le client et le serveur, et de le modifier à sa guise pour l'obtention d'informations (mots de passe, accès système, etc.).

La plupart du temps, l'attaquant utilise les techniques de détournement de flux décrites dans les précédentes sections pour rediriger les flux du clients et du serveur vers lui.



8.2.1. Document

Le document *Man-In-the-Middle Attack* (http://www.giac.org/practical/gsec/Bhavin_Bhansali_GSEC.pdf) est très complet sur cette question.

8.3. Encapsulation d'IP dans d'autres protocoles.

Certains logiciels permettent d'encapsuler le protocole IP dans d'autres protocoles comme SSH, HTTP, etc.. Ce type d'encapsulation peut être la base de nombreuses attaques réseaux.

Par exemple, imaginons cette situation : un pirate veut se connecter sur le port FTP (21) d'une machine A d'un réseau protégé par un firewall B. B n'autorise et n'assure que le trafic HTTP. Si le pirate veut se connecter sur A, il encapsule les paquets à destination de A dans des requêtes HTTP destinées à B. B accepte ces paquets car ils reposent sur le protocole HTTP. Si B est mal configuré, il enverra à A les paquets lui étant destinés.

Chapitre 9. Description d'attaques sur différents protocoles

Introduction

Ce chapitre décrit les failles intrinsèques de différents protocoles. Intrinsèques par le fait qu'elles ne sont pas liées à une faille applicative du client ou du serveur gérant ce protocole, mais plutôt à sa conception. Nous présenterons aussi la manière de s'en protéger.

9.1. Dynamic Host Configuration Protocol - DHCP

Le protocole DHCP est utilisé pour délivrer dynamiquement une adresse IP unique pour chaque machine le demandant sur le réseau interne. En clair, si un client interne veut obtenir une adresse IP pour bénéficier des services réseau, il envoie un message DHCP à tout le réseau (broadcast) pour trouver le serveur DHCP. Le serveur DHCP répondra en lui envoyant tous les paramètres de configuration réseau.

Ce service permet «d'alléger» la gestion du réseau en évitant d'avoir des configurations statiques à maintenir sur chaque machine. Malheureusement, le protocole DHCP comporte diverses failles que nous allons vous présenter.

9.1.1. Attaque par épuisement de ressources

Comme il l'a été décrit, un serveur DHCP possède un stock d'adresses IP qu'il distribue aux différents clients. Ce stock est bien sûr limité. Il y aura seulement un nombre défini de clients pouvant disposer des différentes adresses IP en même temps. Si le serveur est bien administré avec une liste «fermée» de correspondances entre adresses MAC et IP aucune attaque par épuisement n'est possible.

Si le service est mal administré ; c'est à dire que les correspondances entre adresses MAC et IP se font dynamiquement à partir d'une plage d'adresses IP vacantes, le scénario suivant est possible.

Si un pirate génère un grand nombre de requêtes DHCP semblant venir d'un grand nombre de clients différents, le serveur épuisera vite son stock d'adresses. Les «vrais» clients ne pourront donc plus obtenir d'adresse IP : le trafic réseau sera paralysé.

9.1.2. Faux serveurs DHCP

Cette attaque vient en complément de la première. Si un pirate a réussi à saturer un serveur DHCP par épuisement de ressources, il peut très bien en activer un autre à la place. Ainsi il pourra ainsi contrôler tout le trafic réseau.

9.1.3. Comment s'en protéger ?

Chaque fois que c'est possible, il faut limiter le service DHCP à une liste «fermée» de correspondances d'adresses MAC et IP. De cette façon toute requête «étrangère» à cette liste est systématiquement rejetée.

- Sous Windows, remplissez les champs de l'option Réservations dans le programme de configuration du serveur DHCP
- Sous Linux, éditez le fichier `/etc/dhcpd.conf` sur le serveur DHCP. Par exemple, pour un client `toto` avec l'adresse MAC `00:C0:34:45:56:67` à laquelle on fait correspondre : l'adresse `192.168.1.2`, le routeur `192.168.1.1` et le serveur de noms `192.168.1.3`.

```
host toto {
```

```
hardware ethernet 00:C0:34:45:56:67;  
fixed-address 192.168.1.2;  
option routers 192.168.1.1;  
option domain-name-server 192.168.1.3;  
}
```

S'il est impossible d'établir une liste «fermée», segmentez votre réseau en sous-réseaux et attribuez-leur chacun un serveur DHCP. Ces serveurs seront indépendants les uns des autres.

Enfin, les nouvelles versions du protocole DHCP permettent l'utilisation de mécanismes d'authentification plus stricts. Assurez vous que vos serveurs utilisent ces versions de protocoles (Voir RFC3118 (<http://www.faqs.org/rfcs/rfc3118.html>)).

9.1.4. Documents

- Sécurisation sous windows : [FIXIT]
- Sous Linux : *Comment installer un serveur DHCP en 15 minutes grâce à Linux!* (<http://www.digital-connexion.info/article.php?sid=89>) et *How to make Network Configuration as easy as DHCP* (<http://www.linux-mag.com/cgi-bin/printer.pl?issue=2000-04&article=networknirvana>)

9.2. Domain Name Service - DNS

Le protocole DNS assure la correspondance entre le nom d'une machine et son adresse IP. Un serveur DNS est en écoute par défaut sur le UDP port 53. Les attaques décrites ici concernent les faiblesses du protocole DNS.

9.2.1. Le DNS ID spoofing

C'est la première attaque que nous allons décrire. Elle aboutit à un détournement de flux entre deux machines à l'avantage du pirate.

Imaginons qu'un client A veuille établir une connexion avec une machine B. La machine A connaît le nom de la machine B mais pas son adresse IP, ce qui lui empêche pouvoir communiquer avec. La machine A va donc envoyer une requête au serveur DNS du réseau de B pour connaître l'adresse IP de B, cette requête sera identifiée par un numéro d'identification (ID). Le serveur répond à cette requête en fournissant l'adresse IP de B et en utilisant le même numéro d'ID.

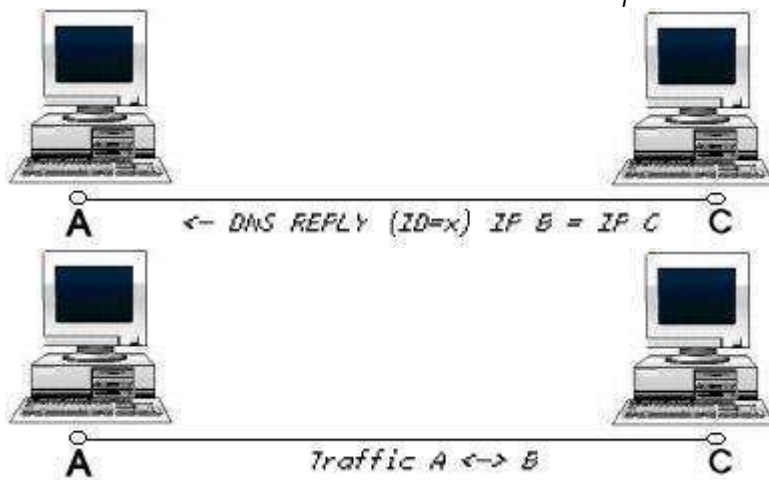
Ce numéro a une valeur comprise entre 0 et 65535.

Le DNS ID spoofing a pour but de d'envoyer une fausse réponse à une requête DNS avant le serveur DNS. De cette façon, le pirate peut rediriger vers lui le trafic à destination d'une machine qu'il l'intéresse.

Dans notre exemple, un pirate C doit répondre à A avant le serveur DNS (D) du réseau de B. Ainsi, il envoie à A son adresse IP associée au nom de la machine B. A communiquera alors avec le pirate C au lieu de la machine B.

Illustration :





Néanmoins, pour implémenter cette attaque, le pirate doit connaître l' ID de requête DNS. Pour cela, il peut utiliser un sniffer s'il est sur le même réseau, soit prédire les numeros d'ID par l'envoi de plusieurs requêtes et l'analyse des réponses.

9.2.2. Le DNS cache poisoning

Le principe de cette attaque est très similaire à celui de l'ARP-Poisoning. Pour gagner du temps dans la gestion des requêtes, le serveur DNS possède un cache temporaire contenant les correspondances adresses IP - noms de machine. En effet, un serveur DNS n'a que la table de correspondance des machines du réseau sur lequel il a autorité. Pour des machines distantes, il doit interroger d'autres serveurs DNS. Pour éviter de les interroger à chaque requête, il garde en mémoire (dans un cache), le résultat des précédentes requêtes.

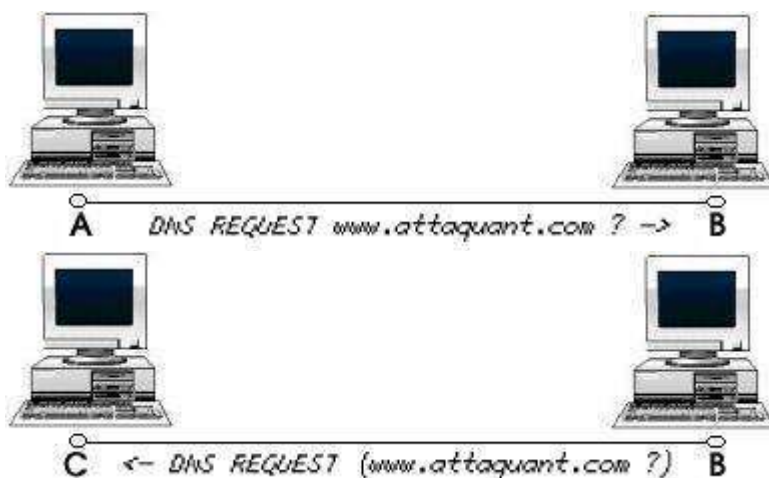
L'objectif du pirate est d'empoisonner ce cache avec de fausses informations. Pour cela, il doit avoir un nom de domaine sous contrôle et son serveur DNS.

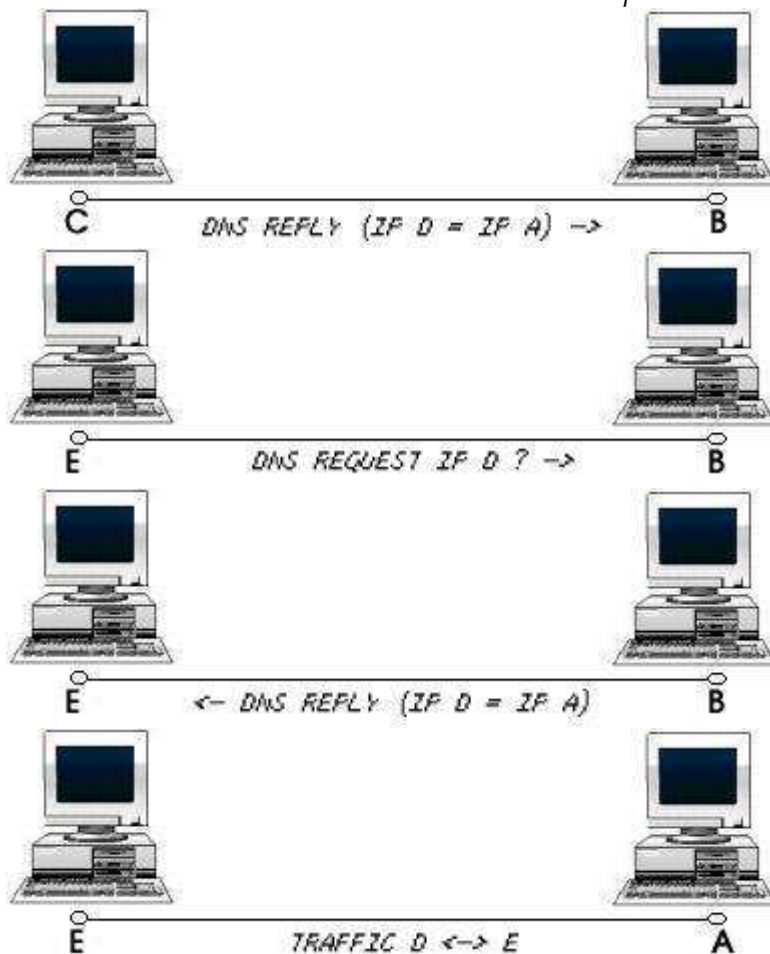
Imaginons qu'un pirate (A) possède le nom de domaine `attaquant.com`, et son serveur DNS (C) et qu'il veuille empoisonner le cache du serveur DNS (B) du réseau `cible.net`.

Le pirate envoie une requête au serveur DNS (B) du réseau `cible.net` demandant la résolution du nom de domaine `attaquant.com`.

Le serveur DNS (B) de `cible.net` va donc envoyer une requête sur le serveur DNS (C) de l'attaquant (c'est lui qui a autorité sur le domaine `attaquant.com`). Celui-ci répondra et joindra des informations additionnelles falsifiées par le pirate (un nom de machine (D) associé à l'adresse IP (A) du pirate). Ces informations seront mises en cache sur le serveur DNS (B) de `cible.net`. Si un client quelconque (E) demande l'adresse IP pour le nom de la machine (D), il recevra l'adresse du pirate (A) en retour.

Illustration :





9.2.3. Comment s'en protéger ?

Configurez votre serveur DNS pour qu'il ne résolve directement que les noms de machine du réseau sur lequel il a autorité.

Autorisez seulement des machines internes à demander la résolution de noms de domaines distants.

Mettez à jour ou changez les logiciels assurant le service DNS pour qu'ils vous protègent des attaques décrites précédemment.

9.2.4. Documents

Sous linux : *DNS et sécurité* (<http://www.toolinux.com/linutile/reseau/dns/index20.htm>) et *DNS et sécurité* (<http://www.linux-france.org/article/memo/dns/node16.html>)

9.3. FINGER

Le service `finger` permet d'obtenir des informations sur les utilisateurs du système.

`finger` s'invoque simplement avec la commande :

```
[root@nowhere /root]#finger @machinecible
```

Le symbole `@` produit le même effet que l'astérisque pour un listing de répertoire. En effet, les informations concernant tous les utilisateurs connectés à la machine de nom `machinecible` seront listées et envoyées en réponse à la requête. Exemple :

```
[root@nowhere /root]#finger @machinecible
```

```
Login Name Tty Idle Login Time Office toto Le toto pts/7 3d Mar 26 20 :43
(case)// root root pts/4 5d May 25 16 :20
```

On voit ainsi qui est connecté sur le système (toto et root) et depuis quand (colonne Time).

finger n'est pas dangereux mais le laisser en écoute, sans en avoir réellement besoin, est une grossière erreur. finger donne trop d'informations sur les utilisateurs systèmes.

9.3.1. Comment s'en protéger ?

- Sous UNIX, il est conseillé de désactiver le service finger dans le fichier /etc/inetd.conf. Pour cela, ajoutez un dièse (#) devant la ligne relative au service finger.

```
# finger stream tcp nowait root /usr/sbin/tcpd in.fingerd
```

- Sous Windows, désactivez le programme associé au service finger.

Si vous ne souhaitez pas désactiver le service finger, configurez votre firewall pour limiter les accès vers ce service.

9.4. FTP

FTP (*File Transfert Protocol*, en écoute par défaut sur les ports 20 et 21) est le service utilisé pour assurer le transfert de fichiers. Il y a deux types de serveurs FTP : les serveurs FTP avec authentification par mots de passe et les serveurs anonymes. Pour les premiers, le client désirant se connecter devra fournir un login accompagné d'un mot de passe pour authentification. Dans le cas du serveur FTP anonyme, tout le monde peut s'y connecter librement.

Le premier défaut du protocole FTP est de ne pas encrypter les mots de passe lors de leur transit sur le réseau. Les mots de passe associés aux logins circulent en clair à la merci des sniffers.

Voici l'exemple d'une interception par un sniffer d'une authentification FTP :

Le logiciel utilisé est tcpdump.

```
22 :10 :39.528557 192.168.1.3.1027 > 192.168.1.4.ftp : P 1 :12(11) ack
47
win 5840 ;nop,nop,timestamp 441749 100314;> (DF) [tos 0x10]
0x0000 4510 003f 88d6 4000 4006 2e7b c0a8 0103 E.. ?..@.@.....
0x0010 c0a8 0104 0403 0015 e351 3262 8d6a dd80 .....Q2b.j..
0x0020 8018 16d0 68da 0000 0101 080a 0006 bd95 ....h.....
0x0030 0001 87da 5553 4552 2061 6c65 780d 0a00 ....0,1,0USER.alex...
```

```
22 :10 :57.746008 192.168.1.3.1027 > 192.168.1.4.ftp : P 12 :23(11) ack
80
win 5840 ;nop,nop,timestamp 443571 101048;> (DF) [tos 0x10]
0x0000 4510 003f 88d8 4000 4006 2e79 c0a8 0103 E.. ?..@.@..y....
0x0010 c0a8 0104 0403 0015 e351 326d 8d6a dda1 .....Q2m.j..
0x0020 8018 16d0 5ba1 0000 0101 080a 0006 c4b3 ....[.....
0x0030 0001 8ab8 5041 5353 2074 6f74 6f0d 0a00 ....0,1,0PASS.toto...
```

On peut voir facilement que l'utilisateur alex a le mot de passe toto.

9.4.1. Le serveur FTP anonyme

Le serveur FTP anonyme pose de plus gros problèmes. Le premier est qu'une mauvaise gestion des droits d'accès peut s'avérer être une erreur fatale. Laisser trop de répertoires en droit d'écriture et/ou d'exécution est plus que dangereux pour la sûreté du système. Le pirate pourrait y installer ou y exécuter des codes malveillants lui permettant d'accroître son pouvoir sur la machine.

9.4.1.1. Boucing attack - Attaque par rebonds

Les serveurs FTP anonymes peuvent être sujets à des attaques par rebonds. Ces attaques consistent à utiliser un serveur FTP anonyme comme relais pour se connecter à d'autres serveurs FTP. Imaginons qu'un pirate se voit refuser l'accès par un serveur FTP dont l'accès est alloué à seulement un certain groupe d'adresses IP. Imaginons que le pirate ne fait pas partie de ce groupe, mais qu'un serveur FTP anonyme y appartienne. Le pirate peut très bien se connecter sur le serveur FTP anonyme, utiliser les commandes assurant la connexion sur le serveur FTP protégé et y récupérer des fichiers.

9.4.2. Comment s'en protéger ?

Installez un serveur FTP anonyme seulement en cas d'absolue nécessité. Si vous devez le faire, limitez au maximum les droits sur les différents répertoires et fichiers laissés au public.

Pour vous protéger des attaques par sniffer, je vous recommande d'utiliser SFTP (Secure FTP) pour vos transactions FTP. SFTP cryptera les échanges et les protégera ainsi des écoutes indiscretes. Vous pouvez aussi utiliser des tunnels comme IPSec pour protéger vos connexions (voir section #x1-460004.3).

Filtrez les accès (via un firewall) en allouant seulement l'accès à un certain groupe d'adresses IP (en évitant d'inclure des serveurs anonymes permettant de servir de relais).

9.5. HTTP

Un serveur HTTP est en écoute sur le port 80. Le protocole HTTP est sûrement le plus utilisé sur le web pour les pages html. Ce protocole ne comporte pas de failles intrinsèques majeures. Par contre, les applications assurant son traitement sont souvent bourrées de failles. Cela vient du fait que le web devient de plus en plus demandeur en terme de convivialité et cela génère une complexité plus grande des applications, d'où un risque de failles plus important.

Nous allons décrire ces failles une à une.

9.5.1. Les serveurs trop bavards

Parfois, les bannières des serveurs web sont trop explicites. Exemple sur un serveur Apache :

```
[root@nowhere /root]# telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 04 Jan 2004 15:07:06 GMT
Server: Apache/1.3.29 (Debian GNU/Linux)
Last-Modified: Sat, 24 Nov 2001 16:48:12 GMT
ETag: "17082-100e-3bffcfc4c"
Accept-Ranges: bytes
Content-Length: 4110
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

Lors de l'envoi de la commande **HEAD / HTTP/1.0**, trop d'informations sont données.

Les pages d'erreurs (404 : page non trouvée) peuvent aussi contenir des informations sur le système.

9.5.2. Vulnérabilités liées aux applications web

La complexité des serveurs ou des navigateurs (clients) web pose de gros problèmes de sécurité. Ces applications sont vulnérables à de nombreux bugs. Chaque application a son type de faille. Netscape par exemple devient vulnérable lors du traitement de certaines chaînes de caractères. Cela peut permettre de remonter toute l'arborescence des fichiers du serveur. Les serveurs IIS peuvent renvoyer un shell système pour un envoi de commandes particulières.

Les langages comme Javascript, Perl, PHP, ASP pour la réalisation de scripts peuvent se révéler dangereux. L'origine d'une faille dans une application web peut apparaître à cause de deux problèmes. Le premier est la fiabilité de la conception du script, le second est la fiabilité des fonctions utilisées. Si un script est mal conçu, il peut être la source de nombreuses failles. De même, si sa conception est bonne mais qu'il utilise des fonctions boguées, il peut se révéler encore plus dangereux.

9.5.3. Comment se protéger ?

Vérifiez que votre serveur web n'est pas trop bavard. Si c'est le cas, modifiez sa configuration pour qu'il se taise. Pour cela, consultez la documentation pour modifier le contenu des messages d'erreur ou de bienvenue.

Un serveur web ne devrait jamais être exécuté avec les droits administrateurs.

Mettez à jour les navigateurs et les serveurs pour prévoir d'éventuelles failles.

Lors du développement de scripts, prenez garde lors de la conception à la gestion des droits des utilisateurs pour son exécution. Informez-vous aussi sur les fonctions connues pour être «sensibles».

Les NIDS peuvent être une bonne parade contre les attaques reposant sur des failles logicielles. Ils permettent de détecter l'exécution de telles attaques (Voir [Section 4.2](#)).

L'utilisation de SHTTP (Secure HTTP) est aussi une bonne parade contre les attaques HTTP.

Une bonne définition de SHTTP est donné par E.Rescorla et A. Schiffman :

"Le protocole SHTTP est une extension de HTTP qui fournit des services de sécurité, applicables indépendamment, qui permettent de garantir la confidentialité, l'authenticité/intégrité, et le non refus d'origine."

SSL ("Secure Socket Layer" pour Netscape) permet de protéger les transactions web, il peut être judicieux de l'utiliser.

9.6. IDENT

Le service `ident` (anciennement appelé `auth`, en écoute sur le port 113) est du même genre que le service `finger`. Il fournit des informations sur les détenteurs de connexions sur le système.

Il convient de le supprimer, s'il n'a aucune utilité.

9.6.1. Comment s'en protéger ?

Sous UNIX, pour désactiver le service `ident`, ajoutez un dièse (#) pour commenter la ligne concernant ce service dans le fichier `/etc/inetd.conf`.

```
# :INFO : Info services
#ident stream tcp wait identd /usr/sbin/identd identd
```

9.7. IP et l'IP-Spoofing

Cette méthode de piratage date un peu. Mais elle demeure légendaire par l'utilisation qu'en a fait Kevin Mitnick en 1995 contre le Supercomputer Center de SanDiego protégé par Tsatumo Shimomura. Néanmoins, cette faille était connue depuis février 1985 comme le montre le rapport *Weakness in the 4.2BSD Unix TCP/IP software* écrit par Robert Morris.

L'IP spoofing se base sur une usurpation d'adresse IP. L'IP spoofing est utilisé lorsque deux hôtes sont en relation de confiance grâce à leurs adresses IP, c'est-à-dire que la seule authentification faite au niveau du serveur consiste en une vérification de l'adresse IP du client.

L'IP spoofing a souvent lieu contre les services rlogin et rsh car leur mécanisme d'authentification est basée sur l'adresse IP. Le principe est simple : dès qu'un client possède une connexion établie sur le serveur avec un mode d'authentification basée sur l'adresse IP, le pirate va essayer de se faire passer pour le client auprès du serveur. Pour cela, il va empêcher le client de dialoguer avec le serveur et répondra à sa place.

L'IP-Spoofing est une attaque concernant un nombre limité de machines. Vous découvrirez pourquoi en lisant la suite.

9.7.1. Un peu de théorie ...

Le protocole IP est non-orienté connexion, il n'assure aucune vérification de la réception des paquets et ne se soucie guère de la façon de les traiter. IP n'assure qu'un routage d'une adresse vers autre. Il est donc facile de duper le routage IP en injectant des paquets falsifiés ayant des adresses IP valides sur le réseau.

Le protocole TCP, quant à lui, assure une fiabilité de la remise des paquets grâce à des numéros de séquences qui les distingue un à un. Chaque paquet TCP possède deux numéros, le numéro de séquence et le numéro d'acquittement. Ces deux nombres sont codés sur 32 bits. Ils sont uniques, afin de ne pas confondre les paquets lors de leurs traitements.

C'est sur ces bases que nous allons décrire l'attaque.

Imaginons que le client A est connecté en rsh sur le serveur B. Le pirate C va tenter de voler la connexion au client.

Il va en premier lieu réduire au silence le client en le saturant avec des attaques tels que le syn-flooding, le déni de service (voir [Section 6.4](#)).

La seconde partie de l'attaque est assez simple. Le pirate envoie une série de demandes de connexion au serveur (paquets TCP avec le flag SYN armé) en utilisant l'adresse de A. Le serveur répond avec une série de paquets d'acquittement (flags SYN et ACK armés). C'est là que réside toute la finesse de l'IP spoofing.

Mais d'abord, quelques rappels sur le protocole TCP. Pour l'établissement d'une connexion TCP, le client envoie un paquet avec un numéro de séquence initial (NS1). Le serveur va répondre avec un paquet d'acquittement ayant son propre numéro de séquence (NS2), mais ayant un numéro d'acquittement (NA1) égal au numéro de séquence initial incrémenté d'une unité ($NA1=NS1+1$). Ensuite le client renvoie un paquet avec un numéro d'acquittement ($NA2=NS2+1$). Une connexion TCP s'établit donc en trois parties.

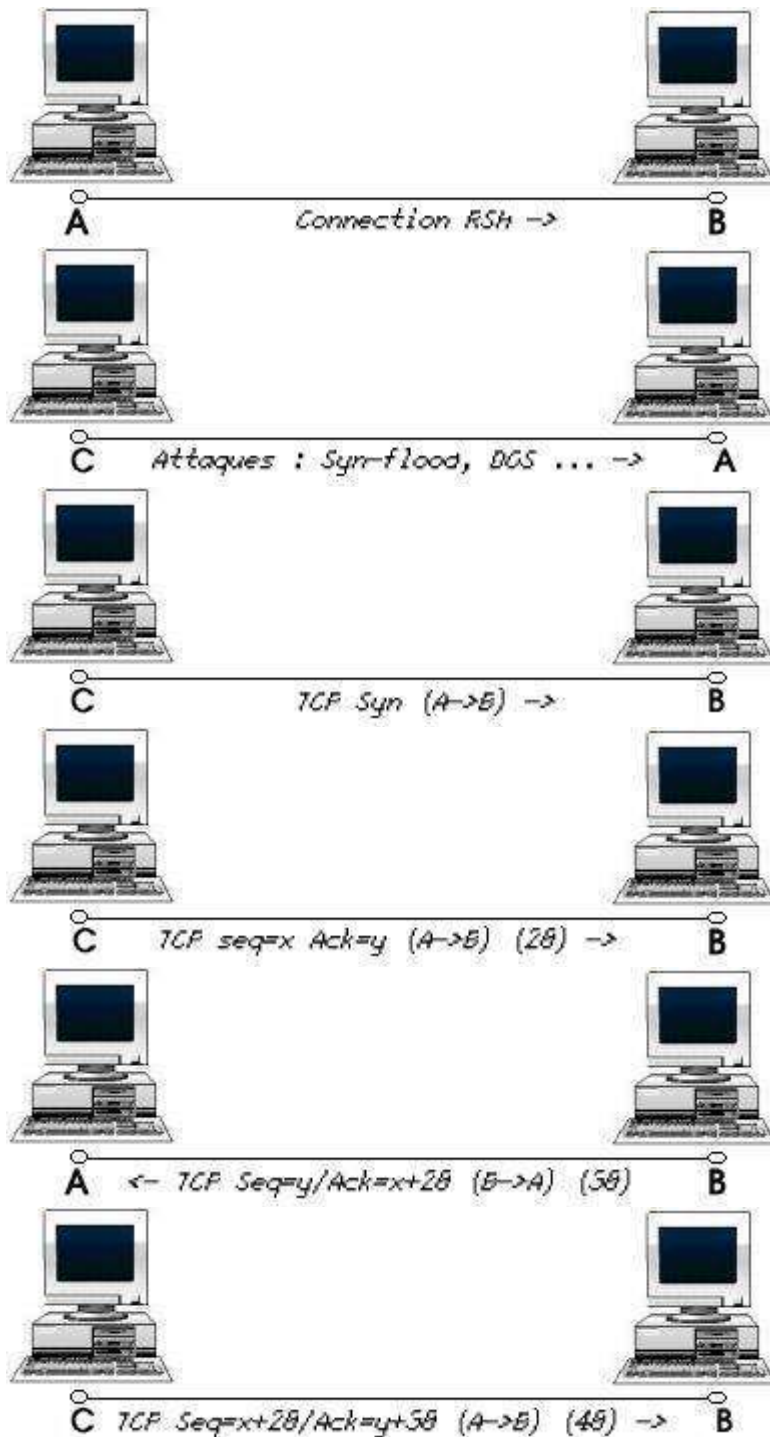
Ce principe de numéros de séquences et d'acquittement est utilisé tout le long de la transaction pour en assurer la fiabilité. La subtilité de l'attaque réside dans le fait que le serveur génère la valeur NS2 suivant un cycle particulier. Il peut utiliser, par exemple, soit une fonction générant un nombre aléatoire, soit incrémenter une valeur initiale de 128 toutes les secondes et de 64 après chaque connexion. Tout dépend de l'implémentation de la pile TCP/IP du système.

Et c'est là que réside tout le problème. Le pirate envoie un grand nombre de demandes de connexion dans un laps de temps déterminé et analyse les acquittements du serveur pour déterminer l'algorithme d'incrémentation. Si cet algorithme est basé sur la génération de nombres aléatoires, l'attaque a peu de chances d'aboutir. Mais si l'algorithme est facilement compréhensible, le pirate va alors envoyer une requête de connexion au serveur en utilisant l'adresse IP du client. Le serveur va répondre avec un paquet d'acquittement de numero de sequence (NS). Le client mis hors service ne pourra répondre, le pirate le fera à sa place.

Pour cela, il doit injecter un paquet ayant un numéro d'acquittement de valeur $NA = NS + 1$. Mais, ayant usurpé l'adresse du client, il ne peut intercepter les paquets lui étant destinés. Il ne peut donc pas connaître cette valeur NS. Il va donc la générer lui-même à partir de son analyse de l'algorithme d'incrémentation. C'est pourquoi cette attaque est aussi qualifiée «d'attaque aveugle». Si le numéro est valide, le pirate a établi la connexion au serveur en se faisant passer pour le client.

Le fait que l'attaque ne se restreigne qu'à une petite partie de systèmes vient du fait que la plupart des piles TCP/IP utilisent des numéros de séquences basés sur des nombres aléatoires. Certains systèmes comme BSD ou HP-UX connaissent de gros problèmes à cause de l'IP-Spoofing.

Illustration





9.7.2. Prévenir l'IP spoofing grâce à Nmap

Nmap invoqué avec l'option `-O` et `-v` vous fournit une indication sur la difficulté qu'aura le pirate à procéder à une attaque par IP spoofing contre votre serveur.

Exemple :

```
[root@nowhere /root]# nmap -O -v 192.168.1.4

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Host (192.168.1.4) appears to be up ... good.
Initiating Connect() Scan against (192.168.1.4)
Adding open port 111/tcp
Adding open port 21/tcp
Adding open port 515/tcp
Adding open port 1024/tcp
Adding open port 22/tcp
Adding open port 139/tcp
The Connect() Scan took 1 second to scan 1554 ports.
For OSScan assuming that port 21 is open and port 1 is closed and neither are firewalled
Interesting ports on (192.168.1.4) :
(The 1548 ports scanned but not shown below are in state : closed)
Port State Service
21/tcp open ftp
22/tcp open ssh
111/tcp open sunrpc
139/tcp open netbios-ssn
515/tcp open printer
1024/tcp open kdm

Remote operating system guess : Linux 2.1.19 - 2.2.19
Uptime 0.122 days (since Thu Mar 27 16 :02 :38 2003)
TCP Sequence Prediction : Class=random positive increments
Difficulty=4687481 (Good luck !)
IPID Sequence Generation : Incremental

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
```

Les lignes intéressantes :

```
TCP Sequence Prediction : Class=random positive increments
Difficulty=4687481 (Good luck !)
```

Celles-ci nous renseignent sur la difficulté d'une attaque par IP-Spoofing. Plus le nombre associé à la valeur `Difficulty` est élevé, plus il est difficile d'entreprendre une attaque. Le message `Good Luck !` entre parenthèses est plutôt ironique vis-à-vis de la réussite de l'attaque.

Si, par malchance, lors d'un scan, vous obtenez un nombre très bas avec un message du type "Trivial Joke", cela signifie que votre système est très vulnérable à une attaque par IP-Spoofing.

9.7.3. Comment s'en protéger ?

Sur la plupart des systèmes, les numéros de séquence sont incrémentés de façon aléatoire, ce qui limite déjà une grande partie des attaques par IP spoofing.

Pour se protéger, il faut commencer par supprimer tous les services se basant sur l'authentification IP (rlogin, rsh).

Certains modules comme `rp_filter` sous Linux permettent une défense contre ces attaques.

L'utilisation de tunnels permet également de parer cette attaque.

9.7.4. Document

Description de l'IP-Spoofing : *IP-spoofing Demystified* (<http://www.phrack.org/show.php?p=48&a=14>).

9.8. NETBIOS

NETBIOS n'est pas un protocole en lui-même, c'est une interface logicielle et un système de nommage. L'interface NETBIOS est très utilisée sur les réseaux Microsoft NETBIOS permet par exemple de partager des ressources en réseau. Ces ressources peuvent être des imprimantes, processus ou des espaces disques. Un pirate peut essayer d'accéder à ces ressources en s'y connectant et tester différents couples utilisateur/mot de passe. NETBIOS n'est pas une interface très sécurisée. Elle est surtout utilisée dans les réseaux Microsoft pour le protocole SMB (bien qu'elle tend à être remplacée).

9.8.1. Comment s'en protéger ?

Protégez toutes vos ressources NETBIOS par mot de passe. Ne laissez jamais un service NETBIOS de votre réseau en écoute sur Internet.

Autorisez seulement l'accès aux ressources NETBIOS au client de votre réseau.

9.8.2. Document

Un bon guide pour la sécurisation NETBIOS (en anglais) : *A STUDY IN REMOTE NT PENETRATION* (<http://packetstormsecurity.org/groups/rhino9/wardoc.txt>)

9.9. Network File System - NFS

NFS a pour but de partager un ensemble de fichiers sur un réseau. Il est souvent couplé à un serveur NIS pour l'authentification.

NFS a été développé par Sun Microsystems.

9.9.1. Les attaques

Le gros problème avec NFS, c'est que le client fait toujours confiance au serveur et vice versa. Un compte "root" sur une machine cliente peut très bien compromettre le serveur et vice-versa.

9.9.2. Comment s'en protéger ?

Utilisez la commande `mount` avec l'option `-nosuid` sur les clients, ce qui empêche la modification ou l'utilisation des programmes `setuid` sur un système de fichier NFS.

Utilisez l'option `root-squash` dans le fichier `/etc/exports`. Cela aura pour but d'empêcher un client `root` de devenir `root` sur le système de fichiers NFS stockés sur le serveur.

Protéger l'accès à votre serveur NFS en filtrant à partir de l'adresse IP, les différents clients.

9.10. Network Information Service - NIS

NIS permet de partager, à travers un réseau, une base de données contenant des bases d'informations pour chacune des machines constituant le réseau (fichier de mots de passe, listes d'hôtes...).

La base de donnée est gérée par un serveur-maître qui la partage avec des serveurs-esclaves pour être accessible aux machines clientes. Cette base de données est identifiée par un nom de domaine propre à NIS.

Ce service a été développé par SUN Microsystems. Il s'accompagne souvent du service NFS pour permettre le partage de fichiers.

9.10.1. Les attaques

Il est possible d'obtenir des nombreuses informations (notamment les fichiers de mots de passe) à partir du nom de domaine NIS.

Il est aussi possible pour un utilisateur non autorisé d'obtenir les fichiers de mots de passe à partir d'un poste local client NIS.

Les mots de passe sont transmis avec un niveau de chiffrement faible sur le réseau, donc facilement interceptable par un sniffer.

9.10.2. Comment s'en protéger ?

Evitez de donner le même nom de domaine DNS au domaine NIS.

Vérifiez si votre version de NIS vous assure une vérification de l'adresse du domaine depuis lequel sont lancées les requêtes.

Supprimez la commande `yppcat` sur les ordinateurs clients.

Protégez l'accès à votre serveur NIS en filtrant à partir de l'adresse IP, les différents clients.

9.11. PORTMAP

portmap (en écoute sur le port 111) est le support de nombreux autres services comme les serveurs NFS, NIS ... La commande `rpcinfo` permet de savoir quels services RPC sont actifs sur le système visé (ici "machinecible").

```
[root@nowhere /root]# rpcinfo -p machinecible
```

```
program vers proto port
10000 2 tcp 111 portmapper
10000 2 udp 111 portmapper
10007 2 udp 661 ypbind
10007 2 tcp 664 ypbind
```

Lors de la requête, portmap ne possède aucun mécanisme de contrôle, il accepte donc la requête et la traite.

9.11.1. Comment s'en protéger ?

Il est conseillé de filtrer l'accès sur ce port grâce à un firewall bien configuré ou de désactiver totalement ce service.

9.12. Le protocole SMB

Les serveurs SMB sont en écoute sur le port 139 ou 445.

SMB (pour Server Message Block) est le protocole utilisé pour interfacier les partages et les authentifications MICROSOFT. Les clients et serveurs SMB sous Linux et d'autres OS libres utilisent SAMBA pour traiter les échanges avec ce protocole.

SMB possède deux modes d'authentification : le mode "share", dans lequel il associe un mot de passe à une ressource (espace disque, imprimantes ...), et le mode "user", où il associe un mot de passe à un utilisateur. Cet utilisateur peut être aussi propriétaire d'une ressource.

SMB utilise aussi deux modes pour l'envoi de ces mots de passe : encryptés ou non. C'est là que réside toute la faille. C'est le serveur qui donne l'information au client s'il supporte l'encryptage ou non.

Si un pirate parvient à détecter un établissement de session SMB avant cet échange, il peut très bien détourner le flux entre les deux et demander au client d'envoyer son mot de passe en clair et le recevoir.

9.12.1. Les scans de SMB shares

Si vous avez des ressources partagées en accès libre à tout le monde (everyone shares), un pirate utilisera un scanner de share pour les détecter et s'y connecter.

Même si vous protégez ces shares par mot de passe, certains logiciels peuvent tester différents mots de passe en se loguant à la ressource et ainsi tenter différentes combinaisons de mots de passe (voir [Chapitre 7](#)).

9.12.2. Comment s'en protéger ?

Pour être sûr que les mots de passe SMB soient envoyés encryptés sur le réseau (même en cas de requêtes «frauduleuses»), vérifiez que la valeur des clés suivantes de la base de registre sont bien égales à 0 :

- Sous Windows NT4 :
`HKEY\LOCAL\MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters
"EnablePlainTextPasswordi"=dword :00000000";`
- Sous Windows XP :
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanworkstation\parameters
"enableplaintextpassword"=dword :00000000"`

Protégez toutes vos ressources par mots de passe, évitez les shares en accès libre à tous le monde et n'oubliez jamais de choisir des mots de passe associés de la bonne façon (voir [Chapitre 7](#)).

Ne laissez jamais un serveur SMB en écoute sur Internet, cela est plus que suicidaire. Si vous êtes obligés d'utiliser SMB à travers Internet, utilisez les tunnels (voir section #x1-460004.3).

Sur votre réseau interne, filtrez l'accès sur votre serveur SMB grâce à un firewall.

9.12.3. Document

La description des problèmes de sécurité sur SMB (en anglais) : *SMB/CIFS BY THE ROOT* (<http://www.phrack.org/show.php?p=60&a=11>)

9.13. Le service de messagerie - SMTP

SMTP : *Simple Mail Transfert Protocol*.

Un serveur SMTP sert à envoyer les mails sur le réseau local ou sur Internet. Ce service est en écoute sur le port 25.

Le premier problème avec un serveur SMTP est qu'il peut servir de relais de mailing anonyme. Un pirate peut très bien s'en servir pour envoyer des mails scabreux à travers Internet.

Un autre problème concerne les commandes **EXPN** et **VERFY**. Ces commandes sont sources de nombreuses informations pour le pirate. Il convient de les désactiver si le logiciel de messagerie le permet.

9.13.1. Comment s'en protéger ?

Appliquez des règles de firewalling assez strictes concernant le serveur SMTP (usage réservé exclusivement aux machines du réseau interne).

Certains serveurs SMTP empêchent le relaying, vérifiez si votre serveur de messagerie supporte cette option.

9.14. SQL

SQL est utilisé pour la gestion de base de données. Il permet l'interconnexion d'une page web avec une base de données à l'aide de scripts.

Ce chapitre va présenter une attaque très connue contre les serveurs SQL appelée *SQL-Injection*.

9.14.1. L'injection SQL ou SQL-Injection

Une requête SQL passe par plusieurs étapes avant d'aboutir.

Les données sont envoyées par le client par l'intermédiaire d'un script sur le serveur web. Il s'ensuit une connexion au serveur SQL, puis l'envoi des données de la requête du client. La requête est exécutée par le serveur SQL. La réponse est reçue par le client et est affichée sous la forme d'une page web.

L'attaque par SQL-Injection consiste à injecter des caractères spéciaux ou des chaînes de caractères particulières dans les requêtes SQL du client. Ces caractères peuvent être interprétés par le serveur SQL comme des commandes permettant d'obtenir un accès sans mot de passe, de récupérer des fichiers, etc..

9.14.2. Comment s'en protéger ?

Pour bien sécuriser votre serveur SQL, vérifiez que tous les comptes possèdent un mot de passe. Sur certaines versions de serveur, les comptes administrateur ou de certains utilisateurs peuvent être accessibles sans mot de passe après une installation.

Se protéger du SQL-Injection n'est pas toujours aisé, il faut surtout être attentif à la programmation des scripts et aux fonctions utilisées.

9.14.3. Document

La technique du "SQL Injection" décrite par son inventeur : *How I hacked PacketStorm* (<http://www.wiretrip.net/rfp/txt/rfp2k01.txt>)

9.15. SSH

SSH vous permet de créer un tunnel chiffré pour vos connexions, il encrypte toutes les échanges. SSH est donc un outil conseillé (pour ne pas dire indispensable) pour les transactions réseau.

Malgré tout, SSH est sujet à différentes attaques, la première étant une attaque de type *Man-In-the-Middle* (voir [Section 8.2](#)). Le pirate se place entre le client et le serveur (avant l'établissement de session SSH) grâce à des méthodes de redirection de flux. Lorsque le client et le serveur s'échangent les clés indispensables pour l'encryptage, il les intercepte et les utilise pour se faire passer pour le client auprès du serveur et pour le serveur auprès du client.

Une autre attaque utilisant une faiblesse du protocole existe. Cette attaque se base sur la génération aléatoire des caractères de bourrage. Ces caractères sont utilisés pour rendre la longueur du paquet multiple de 64 bits. Ces caractères n'étant pas vérifiables, ils peuvent être utilisés pour modifier le bon déroulement de la communication. Cette faille est plus connue sous le nom de *l'exploitation du canal caché*.

Ces attaques ne sont pas très faciles à implémenter.

9.16. TELNET

Le service telnet (en écoute sur le port 23) permet à deux machines distantes de communiquer. Telnet établit deux terminaux virtuels pour les deux machines. Ce type de terminal est semblable à une connexion série. L'utilisateur a l'impression d'être assis devant un terminal de la machine.

telnet repose sur une authentification par login et mot de passe. Malheureusement, telnet possède le même problème que FTP : il n'assure pas la protection des mots de passe contre l'écoute d'un sniffer. Les mots de passe associés du login circulent en clair sur le réseau.

9.16.1. Comment s'en protéger ?

Utilisez plutôt SSH, qui présente les mêmes fonctionnalités que telnet mais permettant en plus de "tunneler" la connexion en encryptant toutes les transactions.

Utilisez IPsec ou des utilitaires de "tunneling" (voir section #x1-460004.3) pour protéger toutes vos connexions.

Je vous conseille aussi de restreindre l'accès à votre serveur telnet via un firewall.

9.17. XWINDOW

XWINDOW est un service permettant la gestion des interfaces graphiques sous UNIX. XWINDOW est en écoute sur le port 6000.

9.17.1. Les attaques

Il faut savoir que XWINDOW fonctionne sur une architecture client-serveur. Il est donc possible à un client connecté sur un serveur X d'interagir avec celui-ci. Il peut agir sur les fenêtres, capturer des événements X ou en créer. Si un serveur X WINDOW est mal configuré, n'importe quel client pourra s'y connecter et modifier son fonctionnement.

9.17.2. Comment s'en protéger ?

Utilisez la commande `xhost` pour fermer les accès au serveur XWINDOW.

```
[root@nowhere /root]# xhost -
```

9.18. Peer To Peer (eDonkey, Kazaa, etc.)

Les applications *Peer to Peer* permettent de faciliter l'échange de fichiers entre particuliers. Ces applications sont nombreuses et diverses (eDonkey, Kazaa, etc.). Malheureusement, leur utilisation dans un réseau peut être source de problèmes de sécurité et consommer inutilement de la bande passante.

9.18.1. Les outils Peer To Peer sont des vecteurs de virus

Les utilisateurs des applications *Peer to Peer* constituent des cibles idéales pour la propagation des virus. Les développements de ces applications sont récents ; ils n'ont pas subi d'audit sérieux. Le «pouvoir d'attraction» de l'échange de fichier est si grand que cet usage s'est répandu très rapidement. On obtient donc un cocktail dangereux : des applications faibles déployées à très grande échelle sur l'Internet.

Les virus récents (postérieur à la série *Gibe-?* de Septembre 2003) ont commencé à utiliser systématiquement les outils *Peer to Peer*. Une fois de plus les utilisateurs ne sont absolument pas conscients de leur «participation» à la propagation de ces virus. Pourtant, si on s'intéresse un peu à la journalisation les traces de propagation sont immédiatement observables. Voici un relevé sur une heure d'exploitation d'un routeur :

```
Security Violations
=====
Feb 17 19:28:14 routeur 14623: Feb 17 20:28:13.155 GMT: %SEC-6-IPACCESSLOGP: list inbound denied tcp
Feb 17 19:33:18 routeur 14625: Feb 17 20:33:17.880 GMT: %SEC-6-IPACCESSLOGP: list inbound denied tcp
Feb 17 19:37:19 routeur 14627: Feb 17 20:37:18.056 GMT: %SEC-6-IPACCESSLOGP: list inbound denied tcp
Feb 17 19:54:20 routeur 14628: Feb 17 20:54:19.200 GMT: %SEC-6-IPACCESSLOGP: list inbound denied tcp
Feb 17 19:57:25 routeur 14629: Feb 17 20:57:24.892 GMT: %SEC-6-IPACCESSLOGP: list inbound denied tcp
Feb 17 20:00:19 routeur 14630: Feb 17 21:00:18.544 GMT: %SEC-6-IPACCESSLOGP: list inbound denied tcp
```

La distinction entre la journalisation d'une utilisation «usuelle» d'un outil *Peer to Peer* et la journalisation de la propagation de virus est immédiate. Pour un échange de fichiers on aurait observé plusieurs dizaines (voir centaines) d'entrées dans le journal sur une heure. Dans le cas présenté, on observe que les adresses IP sources changent pour chaque entrée et que deux applications *Peer to Peer* ont été utilisées : *eDonkey2000* avec les numéros de ports 4662-4663 et *kazaa* avec le numéro de port 1214. Enfin, le «clou du spectacle», le numéro de port destination 3127 correspond à la porte cachée du virus *W32/MyDoom/W32.Novarg.A*.

On retrouve une des caractéristiques des flux réseaux *Peer to Peer* : il est techniquement difficile de limiter ces flux mais ils sont très faciles à observer et à repérer. Si vous aviez encore quelques illusions sur l'utilisation «anonyme» des outils *Peer to Peer*, il est grand temps de les perdre. Il est même probablement trop tard, vous êtes déjà repérés !

9.18.2. Comment s'en protéger ?

Sous linux, le projet *P2PWall - IPTables blocking of P2P traffic* (<http://www.lowth.com/p2pwall/>) donne des utilitaires et des documents permettant d'utiliser le firewall iptables pour filtrer le trafic d'application Peer to Peer.

Chapitre 10. Sécurité avancée

Introduction

Si vous avez lu les précédents chapitres (et surtout le chapitre 4), vous possédez maintenant toutes les connaissances requises pour pouvoir aborder cette partie. Elle concerne l'implémentation et le développement d'outils dédiés à la sécurité. La première section vous montrera comment architecturer son réseau de façon sécurisée et comment bien implémenter les différents outils de protection. La deuxième section vous décrira différentes bibliothèques utilisées pour le développement d'utilitaires réseaux. Nous donnerons en exemple la programmation d'un simple scanner.

10.1. L'architecture sécurisée

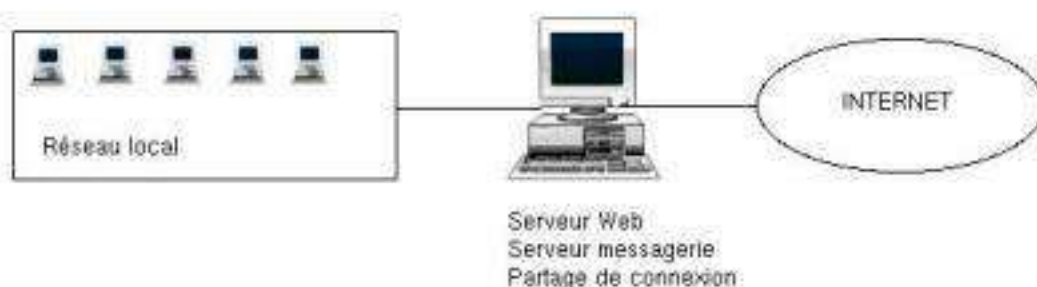
Il existe une infinité de façon d'organiser votre réseau mais, quand la sécurité entre en jeu, il est conseillé d'avoir une architecture réseau bien pensée. Ce chapitre va donner des exemples de réalisation d'architecture sécurisée. Il convient d'avoir lu préalablement les précédents chapitres pour bien saisir la subtilité des architectures décrites.

Nous décrirons différents niveaux d'architecture sécurisée. Nous partirons d'une architecture peu ou pas sécurisée pour arriver à une architecture ultra-sécurisée.

10.1.1. Le réseau de départ

Très simple, une PME ou une université possède un administrateur qui doit gérer l'ensemble du parc informatique et sa conception réseau. Il a pour cahier des charges d'assurer une connexion Internet avec le réseau local, un serveur de messagerie et un site web.

Avec peu de moyens, l'administrateur crée son réseau de la sorte :

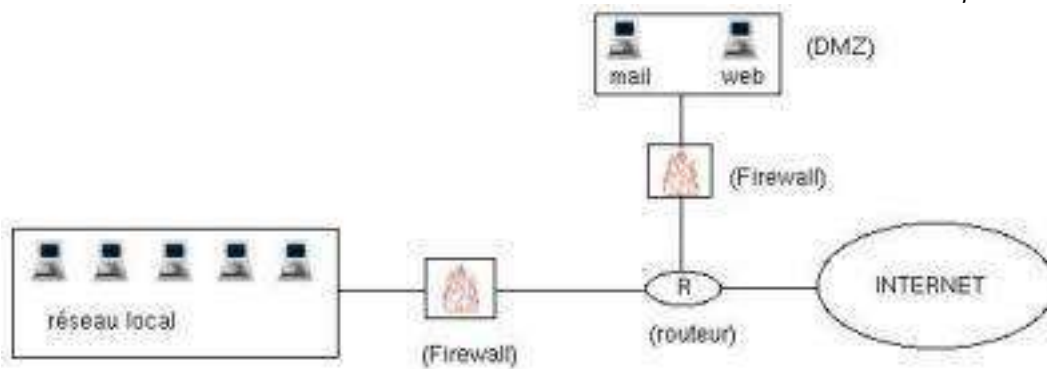


Les serveurs web, de messagerie et de partage de connexion Internet seront assurés par une seule machine. Cette machine est connectée directement à Internet.

Dans ce type de réseau, il n'y a aucune forme de sécurité : la connexion avec Internet n'est absolument pas sécurisée.

10.1.2. Le premier niveau de sécurité

Après un premier piratage, notre administrateur reçoit de nouveaux crédits. Il repense l'architecture :



Il dédie une machine pour le serveur web, une machine pour le serveur de messagerie, deux machines pour le firewalling et un routeur pour assurer la connexion Internet.

Les serveurs de messagerie et web sont dans une zone extérieure à celle du réseau local. Ils constituent une zone démilitarisée (DMZ). Démilitarisée car on peut s'y connecter depuis l'extérieur contrairement au réseau local.

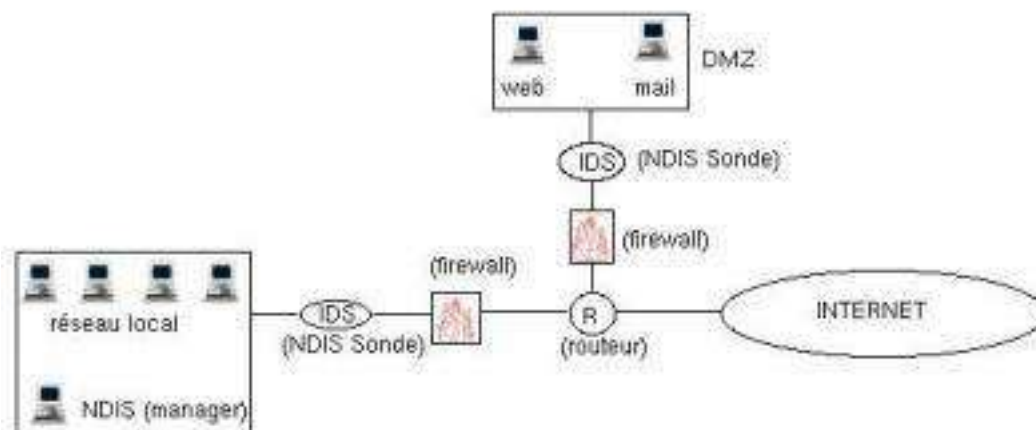
Le firewall situé entre le réseau local et le routeur empêchera toute connexion de l'extérieur vers le réseau local, et autorisera seulement les connexions depuis le réseau local sur un nombre limité de services.

Le firewall situé entre le routeur et la DMZ autorisera tous les accès sur les serveurs web et de messagerie (depuis le réseau local comme depuis l'extérieur), mais en empêchera les tentatives de connexion sur les autres services.

Malheureusement, notre administrateur subit un autre piratage. Il suppose que le pirate a pu passer le routeur et les firewalls en soumettant des requêtes sur des ports autorisés. Il a très bien pu envoyer un exploit sur un serveur web ou de messagerie vulnérable.

10.1.3. Le deuxième niveau de sécurisation

Alors, pour se protéger, il intercale une sonde NDIS (voir [Section 4.2](#)) entre le firewall et le réseau local et dans la DMZ. Si un pirate venait à envoyer des requêtes suspectes ou des exploits connus, les NIDS préviendraient du risque d'intrusion (de l'intérieur ou de l'extérieur). Le manager NDIS se situera dans le réseau local.



Il dédie toujours une seule machine pour un seul service. Il met régulièrement à jour les logiciels, et s'informe sur les nouvelles failles de sécurité.

Ceci constitue un niveau acceptable permettant de résister à des nombreuses attaques.

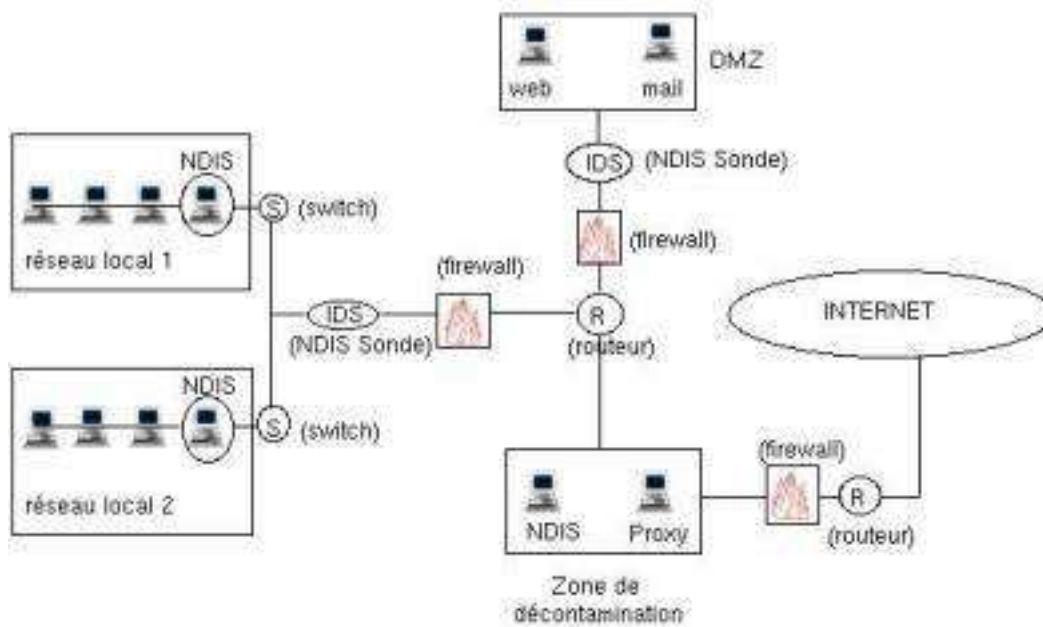
10.1.4. Les niveaux plus élevés

Nous allons nous appuyer dans cette section sur l'architecture précédente. Nous allons modifier certaines parties du réseau.

Nous allons d'abord insérer une zone de décontamination entre Internet et le réseau interne. Cette zone est constituée d'analyseurs de contrôle de contenu, des antivirus et d'autres utilitaires surveillant le trafic réseau (comme des NIDS). Tous les flux entrants et sortants passeront par cette zone de décontamination. Ces proxys applicatifs peuvent prendre la décision de couper la connexion en cas d'attaques ou de simplement rejeter la demande.

Cette zone est appelée zone de décontamination car elle permet de détecter des signatures d'attaques dans les flux de données provenant d'Internet et d'éviter la propagation dans le reste du réseau.

Pour le réseau local, nous le subdiviserons en sous-réseaux, chaque sous-réseau possédera un NDIS (sonde + manager). Ces sous-réseaux seront reliés entre eux par des switches.



Ce type d'architecture est très efficace pour la sécurité, mais reste néanmoins assez coûteuse et difficile à gérer. L'utilisation de tunnels (voir section #x1-460004.3) permet en plus d'accroître la sûreté des transactions.

10.2. Développez vos propres utilitaires sécurité

Ce chapitre va présenter différents outils pour développer vos propres utilitaires de sécurité. Une connaissance du langage C est requise. Une solide connaissance réseau est aussi demandée.

Le système d'exploitation utilisé est Linux avec un compilateur GCC. Nous allons étudier 2 bibliothèques permettant la création d'utilitaires réseaux : libnet et libpcap.

La première, Libnet, est développée par Mike D. Schiffman. C'est une bibliothèque opensource (donc gratuite). Libnet permet de fabriquer et d'injecter facilement des paquets sur un réseau. Un grand nombre de protocoles est supporté.

Note : Note importante : La version de Libnet décrite est la version 1.1. De grosses modifications faites sur le code de la version 1.1 la rend incompatible avec les versions de bibliothèques antérieures.

La deuxième est Libpcap. Elle permet de capturer les paquets transitant sur le réseau. Cette bibliothèque est maintenue par l'équipe de développement de tcpdump (présenté dans le chapitre #x1-690005.4).

Libnet est téléchargeable à cette adresse : www.packetfactory.net

Libpcap est téléchargeable à cette adresse : www.tcpdump.org

Pour donner une description de ces bibliothèques, nous allons étudier le développement d'un scanner de base. Ce scanner listera les ports TCP ouverts sur une machine avec une méthode de scan en port demi-ouvert (SYN scan, voir section #x1-150002.1). L'injection des paquets sera réalisée grâce à Libnet, la réception des réponses grâce à Libpcap.

10.2.1. Le programme

Le programme "scanner" sera appelé en mode console et recevra deux arguments : l'interface réseau d'utilisation et l'adresse IP de la machine à scanner. Il affichera la liste des ports ouverts et se terminera.

Exemple d'utilisation :

```
[root@nowhere.net /root]#./scanner -i eth0 -c 192.168.1.2
```

```
Le port 21 est ouvert
Le port 22 est ouvert
Le port 1111 est ouvert
Le port 1024 est ouvert
```

Mais intéressons-nous à son développement.

Le scan en port demi-ouvert consiste à envoyer un paquet TCP avec le flag SYN armé sur un port précis. Si ce port est ouvert, on recevra en réponse un paquet TCP avec le couple SYN/ACK armé.

Le programme recevra en argument, l'adresse IP de la machine à scanner et le nom de l'interface réseau (par exemple "eth0").

ATTENTION ! Le but de ce chapitre est de présenter les bibliothèques libnet et libpcap et certaines de leurs fonctions les plus utiles d'une manière succincte. Ce chapitre n'est en aucun cas un cours de programmation réseau.

Le principe de fonctionnement du programme est simple ; il sera constitué de deux fonctions : la première permettra d'envoyer les paquets (sur les 16000 premiers ports) et la deuxième de les recevoir. Ces deux fonctions seront activées en parallèle dans deux processus distincts (grâce à l'appel de la fonction fork()).

Je présenterai les deux fonctions utilisées pour intercepter et envoyer :

Pour recevoir (void ReceptionPaquet(unsigned int, u_char device) :

La fonction ReceptionPaquet permet de recevoir les paquets circulant sur le réseau. Pour cela, elle utilise la bibliothèque libpcap. Nous utiliserons deux fonctions de cette bibliothèque pour notre programme :

La première est la fonction pcap_t *pcap_t pcap_open_live(char *device, int snaplen, int promisc,int to_ms,char *errbuf).

Cette fonction initialise la carte réseau et renvoie un descripteur de type pcap_t sur cette interface réseau en écoute.

Le paramètre *device est un pointeur sur une chaîne de caractères contenant le nom de l'interface réseau utilisée (par exemple "eth0").

Le paramètre snaplen représente la taille maximale (en octet) d'un paquet intercepté (max=65535).

Le paramètre promisc contrôle le fonctionnement de la carte réseau. S'il est égal à 1, la carte est en mode transparent, c'est à dire qu'elle intercepte tous les paquets (même ceux qui ne lui sont pas destinés). Si cette valeur est différente de 1, la carte n'acceptera que les paquets lui étant destinés. Pour notre programme, la carte sera en mode transparent donc promisc sera égal à 1.

Le paramètre `to_ms` spécifie le délai (en millisecondes) d'interception de paquets. Lorsque ce délai est écoulé, la fonction se termine.

Le paramètre `*errbuf` est un pointeur sur une chaîne de caractères pouvant contenir un message d'erreur en cas de mauvaise ou non exécution du programme.

La deuxième fonction est la fonction `u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)`. Cette fonction utilise comme argument le descripteur "p" pour accéder à la carte réseau. Elle renvoie chaque paquet intercepté. C'est une fonction bloquante.

Ces deux fonctions sont toujours utilisées en série, la `pcap_open_live` initialise la carte réseau et renvoie un descripteur de fichier, ensuite ce descripteur de fichier est utilisé par la fonction `*pcap_next` qui intercepte les paquets.

Voici la fonction `ReceptionPaquet` :

```
void ReceptionPaquet(unsigned int IP_Cible, u_char *Device)

/*Variable utilisée par les fonctions de libpcap (elle contiendra notre
paquet*/
struct pcap_pkthdr Header ;

/*Descripteur représentant l'interface réseau (retourné par
la fonction pcap_open_live())*/
pcap_t *Descr ;

/*Structure pour l'en tête ETHERNET*/
struct ethhdr *EtherHdr ;

/*Structure pour l'en tête IP*/
struct ip *IpHdr ;

/*Structure pour l'en tête TCP*/
struct tcphdr *TcpHdr ;

/*Tampon pour retour d'erreur*/
char ErrBuf[LIBNET_ERRBUF_SIZE] ;

/*Pointeur vers le paquet*/
u_char *Packet ;

/*Descripteur pour libnet*/
libnet_t *l ;

/* Initialisation pour les fonctions de
libnet (obtention d'un descripteur)*/

l=libnet_init(LIBNET_RAW4,NULL,ErrBuf) ;

/*Descripteur "Descr" sur l'interface réseau en écoute*/

Descr = pcap_open_live(Device,65535,1,0,ErrBuf) ;

while(1) /*On recupere le paquet (il est pointé par la variable
"Packet")*/

    Packet = (u_char *) pcap_next(Descr,&Header) ;

/*On convertit le paquet pour analyser l'en tête ETHERNET*/

    EtherHdr = (struct ethhdr * ) (Packet) ;

/*On verifie si le protocole est bien IP*/

    if(ntohs(EtherHdr->h_proto)==ETH_P_IP)

/*On convertit le paquet pour analyser l'en tête IP*/

        IpHdr = (struct ip * ) (Packet +ETH_HLEN) ;

/*On verifie si le protocole est bien TCP*/

        if(IpHdr->ip_p==IPPROTO_TCP)

            TcpHdr = (struct tcphdr * ) ( Packet + ETH_HLEN +
```



```

4 * (IpHdr->ip_hl)) ;

/* Cela sert à vérifier que nous avons bien envoyé le paquet et qu'il
provient bien de la machine "scannée". Ceci se base sur l'analyse des
adresses IP */

if(

    (IpHdr->ip_src.s_addr==IP_Cible) &&
    (IpHdr->ip_dst.s_addr== libnet_get_ipaddr4(1)))

/*Pour vérifier que le port d'envoi correspond au même que le notre*/
if(ntohs(TcpHdr->dest)==PORT_SOURCE)

    /*Si les flags SYN et ACK sont armés, le port est ouvert*/
if((TcpHdr->ack==1) && (TcpHdr->syn==1))

printf("Le port %d est ouvert
n",ntohs(TcpHdr->source)) ;
    /*Destruction du descripteur*/
libnet_destroy(l) ;

```

Pour envoyer (void EnvoiPaquet(unsigned int,int)) :

La fonction EnvoiPaquet permet d'injecter les paquets sur le réseau. Pour cela, elle utilise la librairie Libnet.

Nous allons utiliser différentes fonctions de la librairie Libnet.

La première est la fonction libnet_t *libnet_init(int injection_type, char *device, char *err_buf).

Cette fonction permet d'initialiser une interface d'injection des paquets. Elle traite trois arguments :

- Le premier injection_type définit le type d'interface (niveau ethernet, IP ou TCP). Pour notre programme, une injection au niveau IP suffira (valeur LIBNET_RAW4).
- Le deuxième argument *device est un pointeur sur la chaîne de caractère contenant le nom de l'interface réseau qui sera sollicitée (ex : eth0), la valeur NULL conviendra car libnet choisira automatiquement une carte réseau.
- Le dernier argument *err_buf est utilisé pour les messages d'erreur (comme libpcap). Elle renvoie un pointeur sur l'interface d'injection de type libnet_t.

Les fonctions libnet_ptag_t libnet_build_tcp(...) et libnet_autobuild_ipv4(...) sont utilisées pour construire les en-têtes TCP et IP. Elles prennent comme arguments les différentes valeurs constituant les en-têtes TCP et IP (numéro de port, de séquences ... pour TCP, adresses IP, types de protocoles pour IP). Elles renvoient toutes les deux un descripteur de type libnet_ptag_t, ces fonctions doivent toujours respecter l'ordre TCP puis IP (sens couche application vers couche réseau ou physique). Ces fonctions prennent en argument le pointeur sur l'interface d'injection (pointeur renvoyé par la fonction libnet_init).

La fonction int libnet_write(libnet_t) injecte le paquet. La fonction void libnet_destroy(libnet_t) détruit l'interface créée par libnet.

Voici la fonction 1,0,0EnvoiPaquet :

```

void EnvoiPaquet (
/*IP Machine Cible*/
unsigned int IP_cible,
/*Port Destination*/
int port_dest)

    /*Variables pour les fonction de libnet*/
char ErrBuf[LIBNET_ERRBUF_SIZE] ;
libnet_t *l ;

libnet_ptag_t Tag ;

    /*Pour initialiser et obtenir un descripteur*/
l=libnet_init(LIBNET_RAW4,NULL,ErrBuf) ;

    /*Pour construire l'en tête TCP*/

```

```

    Tag=libnet_build_tcp(
PORT_SOURCE, /*Port Source*/
port_dest, /*Port destination*/
0, /*N° Séquence*/
0, /*N° Acquitement*/
TH_SYN, /*Demande de connexions*/
4096, /*Taille de fenêtre*/
0, /*Somme de contrôle*/
0, /*Pointeur d'urgence*/
LIBNET_TCP_H, /*Taille en tête*/
(u_char *) (NULL), /*Pointeur vers les données*/
0, /*Taille des données*/
l,
0) ;

    /*Pour construire l'en tête IP*/

    Tag=libnet_autobuild_ipv4(
LIBNET_IPV4_H+LIBNET_TCP_H, /*Taille du paquet*/
IPPROTO_TCP, /*Protocole*/
IP_cible, /*Adresse IP de la machine Cible*/
l) ;

    /*Pour envoyer le paquet*/
libnet_write(l) ;

    /*Pour détruire le descripteur*/

libnet_destroy(l) ;

```

La fonction main() :

La fonction main() traite les arguments de la ligne de commande, lance un processus enfant dans lequel, elle utilisera la fonction ReceptionPaquet et attendra une seconde pour le lancement du processus et de la fonction ReceptionPaquet puis exécutera 16000 fois la boucle d'envoi de paquets (16000 ports). Elle attendra encore 5 secondes pour le traitement des réponses et terminera le programme.

Voici la fonction main :

```

extern char *optarg ;
extern int optind ;
extern int opterr ;

int main(int argc, char *argv[])

    /*Pour avoir les paramètres de la ligne de commande*/
static char optstring[]="i :c : " ;
int optch ;
char *Device ;

    /*Variable d'itération*/
int i ;

    /*Pour stocker l'adresse IP*/
u_int32_t IP_Cible ;

    /*Variable qui va recevoir le PID du processus enfant*/
int Pid ;

    /*Pour traiter les paramètres de la ligne de commande*/

if(argc > 5)

printf("
nscanner -i interface -c IP Cible
n") ;
return 0 ;

    while((optch= getopt(argc,argv,optstring)) !=EOF)

switch(optch)

/*Pour le nom de l'interface*/
case 'i' :

```

```

Device = (char *) (malloc(strlen(optarg)*sizeof(char))) ;
strcpy(Device,optarg,strlen(optarg)) ;
break ;

    /*Pour l'adresse IP de la machine cible*/
case 'c' :
    IP_Cible = inet_addr(optarg) ;
    break ;

    default :
    printf("
nscanner -i interface -c IP Cible
n") ;
    return 0 ;

    /*On lance le processus enfant (récupération et analyse des paquets*/
    Pid=fork() ;

    if(Pid==0)

ReceptionPaquet(IP_Cible,Device) ;

    /*On attend une seconde*/

    sleep(1) ;

    /* On envoie les paquets sur les 16000 premiers ports*/
    for(i=0 ;i<16000 ;i++)
EnvoiPaquet(IP_Cible,i) ;

    /*On détruit le processus enfant*/

    sleep(5) ;

    kill(Pid,SIGINT) ;

    return 0 ;

}

```

10.2.2. Comment obtenir et compiler le source entier du programme ?

Pour le code source du programme et les commandes de compilation, consultez le lien : <http://guidesecu.ifrance.com/prog/scanner.c>

10.2.3. Documents

Pour libnet : <http://www.packetfactory.net>

Pour libpcap : <http://www.tcpdump.org>

Annexe A. Annexes

A.1. Les sites et revues à consulter régulièrement

- CERT (<http://www.cert.org>)
- SecurityFocus (<http://www.securityfocus.com/>)
- *.[packet storm security].* (<http://packetstormsecurity.nl/>)
- Archives Bugtraq (http://citadelle.intrinsec.com/ mailing/ current/HTML/ml_bugtraq/) & Bugtraq France (<http://www.bugtraq-france.com>)
- Liste de diffusion du CNRS sur les virus (<http://www.services.cnrs.fr/wws/info/sos-virus>) & Informations sécurité informatique du CNRS (<http://www.cnrs.fr/Infosecu/Virus.html>)
- CERT RENATER (http://www.renater.fr/Securite/CERT_Renater.htm)
- Phrack : *...a Hacker magazine by the community, for the community...* (<http://www.phrack.org/>)

Pour les revues en français disponibles en kiosque, je vous conseille d'acheter le magazine MISC. Je ne suis pas dépendant de MISC, mais c'est le seul magazine entièrement dédié à la sécurité apportant des réponses concrètes à de nombreux problèmes. Il est disponible tous les 2 ou 3 mois environ chez votre libraire.

Les anciens numéros peuvent être aussi commandés sur le site de MISC, 10 numéros ayant été publié jusqu'en Décembre 2003. Chaque numéro constitue une très bonne source d'informations.

Pour plus de renseignements, consulter *MISC le mag de la sécurité informatique !* (<http://www.miscmag.com/>).

A.2. Remerciements

Je tiens à remercier Nicolas Buratto et Christian Duclou pour leur aide sur ce manuel.

Un grand merci au Mirabellug.

Un petit coucou à celui sans qui rien n'aurait pu être possible.

Pour le soutien musical et moral : Jimi, Timothy, Hunter, Tom, Neil... et les tous les autres...

When the going gets weird, the weird turns pro