LinuxFocus article number 282
http://linuxfocus.org

Les attaques externes

par Eric Detoisien
<valgasu(at)club-internet.fr>

L´auteur:

Eric Detoisien est spécialiste en sécurité informatique. Passionné par tous les domaines liés à la sécurité, il fait partie des experts du groupe rstack - www.rstack.org -

Résumé:

Cet article a été publié dans un numéro spécial sur la sécurité de Linux Magazine France. L'éditeur, les auteurs, les traducteurs ont aimablement accepté que tous les articles de ce numéro hors-série soient publiés dans LinuxFocus. En conséquence, LinuxFocus vous "offrira" ces articles au fur et à mesure de leur traduction en Anglais. Merci à toutes les personnes qui se sont investies dans ce travail. Ce résumé sera reproduit pour chaque article ayant la même origine.

Cet article présente les différentes attaques externes qu'un pirate peut utiliser à l'encontre des machines d'un réseau. Nous aborderons ce sujet au travers des principales attaques réseaux, des attaques via les applications et des attaques de type déni de service.

Les attaques réseaux

Les attaques réseaux s'appuient sur des vulnérabilités liées directement aux protocoles ou à leur implémentation. Il en existe un grand nombre. Néanmoins, la plupart d'entre elles ne sont que des variantes des cinq attaques réseaux les plus connues aujourd'hui.

Fragments attacks

Cette attaque outrepasse la protection des équipements de filtrage IP. Pour sa mise en pratique, les pirates utilisent deux méthodes : les Tiny Fragments et le Fragment Overlapping. Ces attaques étant historiques, les pare-feux actuels les prennent en compte depuis longtemps dans leur implémentation.

Tiny Fragments

D'après la RFC (*Request For Comment*) 791 (IP), tous les noeuds Internet (routeurs) doivent pouvoir transmettre des paquets d'une taille de 68 octets sans les fragmenter d'avantage. En effet, la taille minimale de l'en-tête d'un paquet IP est de 20 octets sans options. Lorsqu'elles sont présentes, la taille maximale de l'en-tête est de 60 octets. Le champ IHL (*Internet Header Length*) contient la longueur de l'en-tête en mots de 32 bits. Ce champ occupant 4 bits, le nombre de valeurs possibles vaut de 2^4 - 1 = 15 (il ne peut pas prendre la valeur 0000). La taille maximale de l'en-tête est donc bien 15*4 = 60 octets. Enfin, le champ *Fragment Offset* qui indique le décalage du premier octet du fragment par rapport au datagramme complet est mesurée en blocs de 8 octets. Un fragment de données occupe donc au moins 8 octets. Nous arrivons bien à un total de 68 octets.

L'attaque consiste à fragmenter sur deux paquets IP une demande de connexion TCP. Le premier paquet IP de 68 octets ne contient comme données que les 8 premiers octets de l'en-tête TCP (ports source et destination ainsi que le numéro de séquence). Les données du second paquet IP renferment alors la demande de connexion TCP (flag SYN à 1 et flag ACK à 0).

Or, les filtres IP appliquent la même règle de filtrage à tous les fragments d'un paquet. Le filtrage du premier fragment (Fragment Offset égal à 0) déterminant cette règle elle s'applique donc aux autres (Fragment Offset égal à 1) sans aucune autre forme de vérification. Ainsi, lors de la défragmentation au niveau IP de la machine cible, le paquet de demande de connexion est reconstitué et passé à la couche TCP. La connexion s'établit alors malgré le filtre IP.

Les figures 1 et 2 montrent les deux fragments et la figure 3 le paquet défragmenté au niveau de la machine cible :

Fig.1: Fragment 1

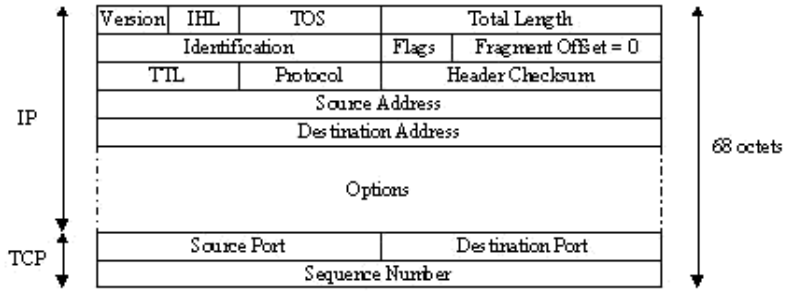


Fig.2: Fragment 2

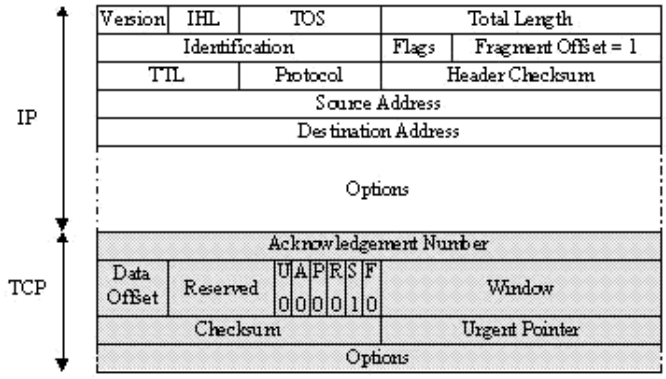
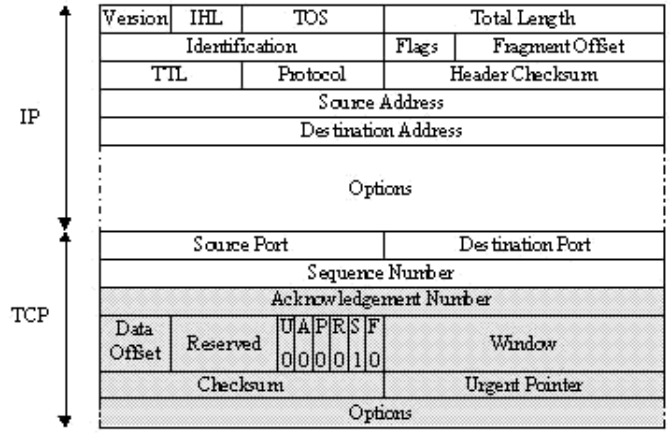


Fig.3: Paquet défragmenté



Fragment Overlapping

Toujours d'après la RFC 791 (IP), si deux fragments IP se superposent, le deuxième écrase le premier. L'attaque consiste à forger deux fragments d'un paquet IP. Le filtre IP accepte le premier de 68 octets (voir Tiny Fragments) car il ne contient aucune demande de connexion TCP (flag SYN = 0 et flag ACK = 0). Cette règle d'acceptation s'applique, là encore, aux autres fragments du paquet. Le deuxième (avec un Fragment Offset égale à 1) contenant les véritables données de connexion est alors accepté par le filtre IP. Ainsi, lors de la défragmentation les données du deuxième fragment écrasent celles du premier à partir de la fin du 8ème octet (car le fragment offset est égal à 1). Le paquet réassemblé constitue donc une demande de connexion valide pour la machine cible. La connexion s'établit malgré le filtre IP.

Les figures 4 et 5 montrent les deux fragments et la figure 6 le paquet défragmenté au niveau de la machine cible :

Fig.4: Fragment 1

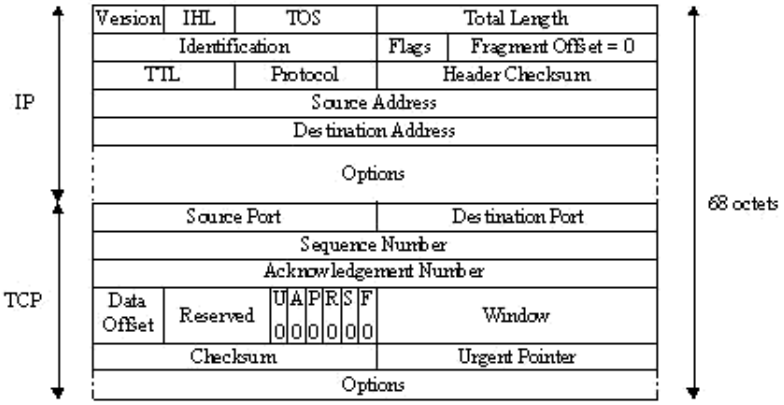


Fig.5: Fragment 2

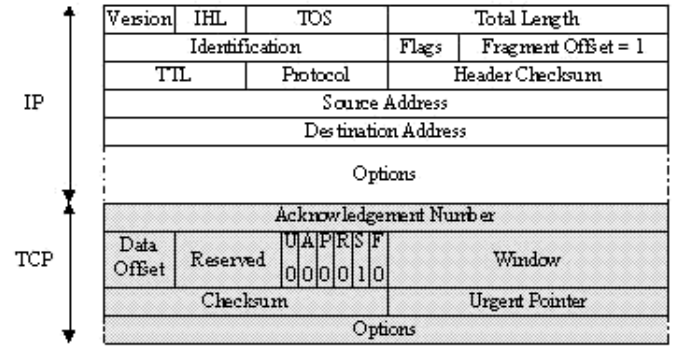
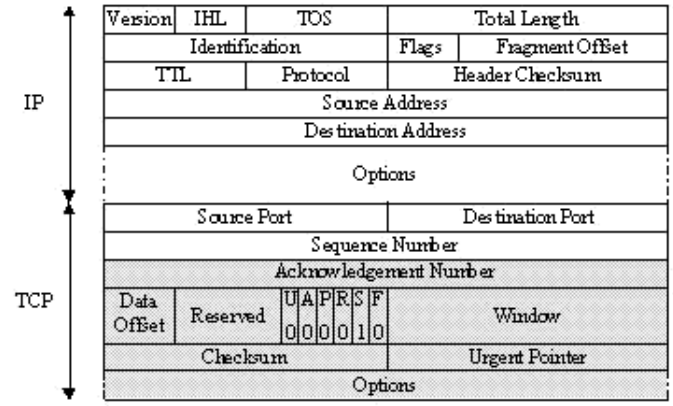


Fig.6: Paquet défragmenté



IP Spoofing

Le but de cette attaque est l'usurpation de l'adresse IP d'une machine. Ceci permet au pirate de cacher la source de son attaque (utilisée dans les dénis de services dont nous discuterons plus tard) ou de profiter d'une relation de confiance entre deux machines. Nous expliquerons donc ici cette deuxième utilisation de l'IP Spoofing.

Le principe de base de cette attaque consiste à forger ses propres paquets IP (avec des programmes comme `hping2` ou `nemesis`) dans lesquels le pirate modifiera, entre autres, l'adresse IP source. L'IP Spoofing est souvent qualifié d'attaque aveugle (ou Blind Spoofing). Effectivement, les réponses éventuelles des paquets envoyés ne peuvent pas arriver sur la machine du pirate puisque la source est falsifiée. Ils se dirigent donc vers la machine spoofée. Il existe néanmoins deux méthodes pour récupérer des réponses :

1. le *Source Routing* : le protocole IP possède une option appelée Source Routing autorisant la spécification du chemin que doivent suivre les paquets IP. Ce chemin est constitué d'une suite d'adresses IP des routeurs que les paquets vont devoir emprunter. Il suffit au pirate d'indiquer un chemin, pour le retour des paquets, jusqu'à un routeur qu'il contrôle. De nos jours, la plupart des implémentations des piles TCP/IP rejettent les paquets avec cette option;
2. le *Reroutage* : les tables des routeurs utilisant le protocole de routage RIP peuvent être modifiées en leur envoyant des paquets RIP avec de nouvelles indications de routage . Ceci dans le but de rerouter les paquets vers un routeur que le pirate maîtrise.

Ces techniques ne sont plus (ou difficilement) utilisables : l'attaque est donc menée sans avoir connaissance des paquets émis par le serveur cible.

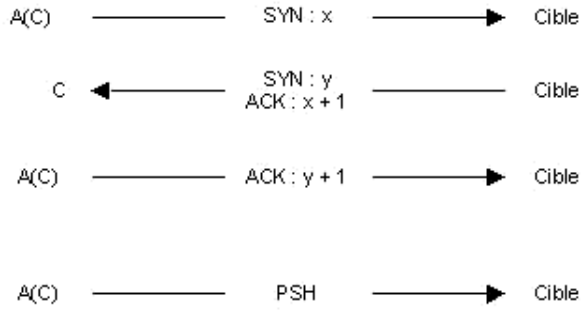
Le Blind Spoofing s'utilise contre des services de type rlogin ou rsh. En effet, leur mécanisme d'authentification se base uniquement sur l'adresse IP source de la machine cliente. Cette attaque relativement bien connue (surtout grâce à Kevin Mitnick qui l'avait utilisée contre la machine de Tsutomu Shimomura en 1994) se déroule en plusieurs étapes :

- détermination de l'adresse IP de la machine de confiance en utilisant par exemple `showmount -e` qui montre où sont exportés les systèmes de fichiers ou `rpcinfo` qui apporte des informations supplémentaires;
- mise hors service de l'hôte de confiance via un SYN Flooding par exemple (nous aborderons les dénis de service plus loin dans cet article). Cela est nécessaire pour que la machine ne puisse pas répondre aux paquets envoyés par le serveur cible. Dans le cas contraire elle enverrait des paquets TCP RST qui mettraient fin à l'établissement de la connexion;
- prédiction des numéros de séquence TCP : à chaque paquet TCP est associé un numéro de séquence initiale. La pile TCP/IP du système d'exploitation le génère de manière linéaire, dépendante du temps, pseudo-aléatoire ou aléatoire selon les systèmes. Le pirate peut uniquement appliquer cette attaque à des systèmes générant des numéros de séquence prévisibles (génération linéaire ou dépendante du temps);
- l'attaque consiste à ouvrir une connexion TCP sur le port souhaité (rsh par exemple). Pour une meilleure compréhension, nous allons rappeler le mécanisme d'ouverture d'une connexion TCP. Elle s'effectue en trois phases (TCP Three Way Handshake) :
    1. l'initiateur envoie un paquet comportant le flag TCP SYN et un numéro de séquence *x*, est envoyé à la machine cible;
    2. cette dernière répond avec un paquet dont les flag TCP SYN et ACK (avec un numéro d'acquittement de *x+1*) sont activés. Son numéro de séquence vaut *y*;
    3. l'initiateur renvoie un paquet contenant le flag TCP ACK (avec un numéro d'acquittement de *y+1*) à la machine cible.

Lors de l'attaque le pirate ne reçoit pas le SYN-ACK envoyé par la cible. Pour que la connexion puisse s'établir, il prédit le numéro de séquence *y* afin d'envoyer un paquet avec le bon numéro de ACK (*y+1*).

La connexion s'établit alors grâce à l'authentification par l'adresse IP. Le pirate peut donc envoyer une commande au service rsh pour obtenir des droits supplémentaires, comme `echo ++ >> /.rhosts`. Pour cela il forge un paquet avec le flag TCP PSH (*Push*) : les données reçues sont immédiatement transmises à la couche supérieure, ici le service rsh, pour que celle-ci les traitent. Il lui est alors possible de se connecter sur la machine directement via un service de type rlogin ou rsh sans IP Spoofing.

La figure 7 résume les étapes de l'IP Spoofing :
Pic.7: IP Spoofing appliqué au service rsh



Le pirate utilise la machine A tandis que la C représente la machine de confiance. La notion A(C) signifie que le paquet est envoyé par A avec l'adresse IP Spoofée de C. A noter l'existence du programme `mendax` qui met en oeuvre ces différents mécanismes de l'IP Spoofing.

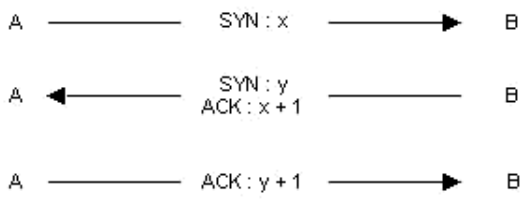
TCP Session Hijacking

Le TCP Session Hijacking permet de rediriger un flux TCP. Un pirate peut alors outrepasser une protection par un mot de passe (comme telnet ou ftp). La nécessité d'une écoute passive (sniffing) restreint le périmètre de cette attaque au réseau physique de la cible. Avant de détailler cette attaque, nous expliquerons quelques principes fondamentaux du protocole TCP.

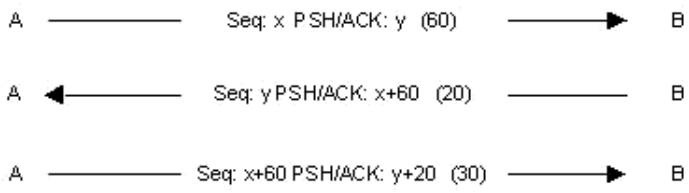
Nous ne dévoilerons pas ici les arcanes du protocole TCP, mais préciserons uniquement les points essentiels à la compréhension de l'attaque. L'en-tête TCP est constitué de plusieurs champs :

- le port source et le port destination, pour identifier la connexion entre deux machines;
- le numéro de séquence qui identifie chacun des octets envoyés;
- le numéro d'acquittement qui correspond au numéro d'acquittement du dernier octet reçu;
- les flags, avec ceux qui vont nous intéresser sont :
  - SYN qui synchronise les numéros de séquence lors de l'établissement d'une connexion;
  - ACK, le flag d'acquittement d'un segment TCP;
  - PSH qui indique au récepteur de remonter les données à l'application.

La figure 8 schématise l'établissement d'une connexion TCP (Three Way Handshake) :
Fig.8 : Three Way Handshake



Dans ce cas, il s'agit de la machine A qui a initialisé une connexion TCP sur la machine B. De même, la figure 9 montre un transfert de données TCP :
Fig.9 : Échange de données:



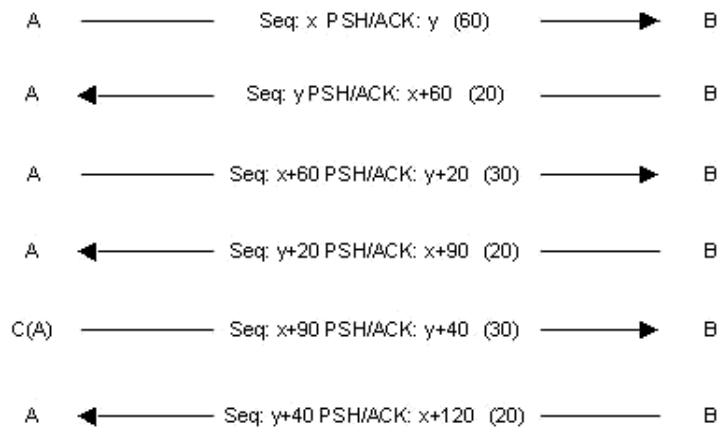
Les numéros de séquence vont évoluer en fonction du nombre d'octets de données envoyés. Le numéro de séquence est représenté par *Seq*, le numéro d'acquittement se trouve après les flags PSH et ACK et le nombre d'octets de données envoyé se trouve entre parenthèses.

Cette attaque crée un état de désynchronisation de chaque côté de la connexion TCP, rendant possible le vol de session. Une connexion est désynchronisée lorsque le numéro de séquence du prochain octet envoyé par la machine A est différent du numéro de séquence du prochain octet à recevoir par B. Et réciproquement, il y a désynchronisation lorsque le numéro de séquence du prochain octet envoyé par la machine B est différent du numéro de séquence du prochain octet à recevoir par A.

Dans l'exemple de la figure 9, à la fin de la première étape quand B reçoit son paquet, A attends un paquet avec un numéro d'acquittement de *x+60*. Si le prochain paquet qu'envoie B n'a pas ce numéro d'acquittement alors A et B sont dits désynchronisés.

Concrètement, un pirate avec une machine C veut voler une session Telnet établie entre les machines A et B. Dans un premier temps, la machine C sniffe le traffic Telnet (port TCP 23) entre A et B. Une fois que le pirate estime que A a eu le temps de s'authentifier auprès du service Telnet de la machine B, il désynchronise la machine A par rapport à B. Pour cela, il forge alors un paquet avec, comme adresse IP source, celle de la machine A et le numéro d'acquittement TCP attendu par B. La machine B accepte donc ce paquet. En plus de désynchroniser la connexion TCP ce paquet permet au pirate d'injecter une commande via la session Telnet préalablement établie par A. En effet, ce paquet peut transporter des données (flag PSH égal à 1).

La figure 10 résume cette attaque :
Fig.10: TCP Session Hijacking



La machine B accepte bien la commande envoyée par C, elle acquitte donc ce paquet en envoyant un paquet à A avec le flag ACK. Entre temps, si A a envoyé un paquet à B celui-ci n'a pas été accepté du fait de numéro de séquence qui n'est pas celui attendu par B.

Un problème apparaît alors : le Ack Storm. Il s'agit d'une multitude de ACK qui sont générés. Ils apparaissent lorsque A envoie un paquet TCP avec un numéro de séquence non valide (car A est désynchronisé) B le jette et envoie à A un ACK avec le numéro de séquence qu'il attend. De son côté, A reçoit ce ACK et comme le numéro de séquence ne correspond pas à celui attendu il renvoie à son tour un ACK à B et B refait la même chose...

Ce problème du Ack Storm peut être réglé si le pirate utilise l'ARP Spoofing. Dans ce cas, la machine C empoisonnera le cache ARP de la machine B en lui indiquant que l'adresse IP de A est désormais associée à l'adresse MAC de C. Ces différentes techniques sont implémentées par le programme `hunt`.

ARP Spoofing

Cette attaque, appelée aussi ARP Redirect, redirige le trafic réseau d'une ou plusieurs machine vers la machine du pirate. Elle s'effectue sur le réseau physique des victimes. Au préalable nous ferons un rappel sur l'utilité et le fonctionnement du protocole ARP.

Le protocole ARP (*Address Resolution Protocol*) implémente le mécanisme de résolution d'une adresse IP en une adresse MAC Ethernet. Les équipements réseaux communiquent en échangeant des trames Ethernet (dans le cas d'un réseau Ethernet bien sûr) au niveau de la couche liaison de données. Pour pouvoir échanger ces informations il est nécessaire que les cartes réseau possèdent une adresse unique au niveau Ethernet, il s'agit de l'adresse MAC (*Media Access Control*).

Quand un paquet IP doit être envoyé la machine expéditrice a besoin de l'adresse MAC du destinataire. Pour cela une requête ARP en broadcast est envoyée à chacune des machines du réseau physique local. Cette requête pose la question : "Quelle est l'adresse MAC associée à cette adresse IP ?". La machine ayant cette adresse IP répond via un paquet ARP, cette réponse indiquant à la machine émettrice l'adresse MAC recherchée. Dès lors, la machine source possède l'adresse MAC correspondant à l'adresse IP destination des paquets qu'elle doit envoyer. Cette correspondance sera gardée pendant un

certain temps au niveau d'un cache (pour éviter de faire une nouvelle requête à chaque paquet IP envoyé).

Cette attaque corrompt le cache de la machine victime. Le pirate envoie des paquets ARP réponse à la machine cible indiquant que la nouvelle adresse MAC correspondant à l'adresse IP d'une passerelle (par exemple) est la sienne. La machine du pirate recevra donc tout le trafic à destination de la passerelle, il lui suffira alors d'écouter passivement le trafic (et/ou le modifier). Il routera ensuite les paquets vers la véritable destination.

L'ARP Spoofing sert dans le cas où le réseau local utilise des switchs. Ceux-ci redirigent les trames Ethernet sur des ports différents selon l'adresse MAC. Il est dès lors impossible à un sniffer de capturer des trames au-delà de son brin physique. L'ARP Spoofing permet ainsi d'écouter le trafic entre des machines situées sur des brins différents au niveau du switch.

Pour mettre en oeuvre une attaque par ARP Spoofing, le pirate va utiliser un générateur de paquet ARP comme `ARPSpoof` ou `nemesis`. Soit la machine victime `10.0.0.171`, sa passerelle par défaut `10.0.0.1` et la machine du pirate `10.0.0.227`. Avant l'attaque un traceroute donne comme résultat :

```
[root@cible -> ~]$ traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.0.1  (10.0.0.1)  1.218 ms  1.061 ms  0.849 ms
```

Et le cache ARP de la machine cible est :

```
[root@cible -> ~]$ arp
Address          HWtype  HWAddress               Flags Mask      Iface
10.0.0.1         ether   00:b0:c2:88:de:65       C               eth0
10.0.0.227       ether   00:00:86:35:c9:3f       C               eth0
```

Le pirate lance alors `ARPSpoof` :

```
[root@pirate -> ~]$ arpspoof -t 10.0.0.171 10.0.0.1
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
```

Les paquets envoyés sont des paquets ARP empoisonnant le cache ARP de la machine `10.0.0.171` avec des ARP Reply indiquant que l'adresse MAC associée à `10.0.0.1` est maintenant `00:00:86:35:c9:3f`.

Désormais, le cache ARP de la machine `10.0.0.171` est :

```
[root@cible -> ~]$ arp
Address          HWtype  HWAddress               Flags Mask      Iface
10.0.0.1         ether   00:00:86:35:c9:3f       C               eth0
10.0.0.227       ether   00:00:86:35:c9:3f       C               eth0
```

Pour vérifier que le trafic passe maintenant par la machine `10.0.0.227` il suffit de faire un nouveau traceroute vers la passerelle `10.0.0.1` :

```
[root@cible -> ~]$ traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.0.227  (10.0.0.227)  1.712 ms  1.465 ms  1.501 ms
 2  10.0.0.1  (10.0.0.1)  2.238 ms  2.121 ms  2.169 ms
```

Le pirate est désormais capable de sniffer le trafic de la machine `10.0.0.171` vers `10.0.0.1`. Il ne faut pas que le pirate oublie d'activer le routage IP sur sa machine `10.0.0.227` (avec l'IP Forwarding - activer la clé `net.ipv4.ip_forward` dans `/etc/sysctl.conf` ou `fragrouter`).

DNS Spoofing

Le protocole DNS (*Domain Name System*) a pour rôle de convertir un nom de domaine (par exemple www.test.com) en son adresse IP (par exemple `192.168.0.1`) et réciproquement, à savoir convertir une adresse IP en un nom de domaine. Cette attaque consiste à faire parvenir de fausses réponses aux requêtes DNS émisent par une victime. Il existe deux méthodes principales pour effectuer cette attaque.

DNS ID Spoofing

L'en-tête du protocole DNS comporte un champ identification qui permet de faire correspondre les réponses aux demandes. L'objectif du DNS ID Spoofing est de renvoyer une fausse réponse à une requête DNS avant le serveur DNS. Pour cela il faut prédire l'ID de la demande. En local, il est simple de le prédire en sniffant le réseau. Néanmoins, cela s'avère plus compliqué à distance. Cependant il existe plusieurs méthodes :

- essayer toutes les possibilités du champ ID. Cette méthode n'est pas très réaliste puisqu'il y a 65535 possibilités pour le champ ID (car ce champ est codé sur 16 bits);
- envoyer quelques centaines de requêtes DNS dans l'ordre. Cette méthode est bien évidemment peu fiable;
- trouver un serveur qui génère des ID prévisibles (incrémentation de 1 de l'ID par exemple), ce genre de faille existe sur certaines version de Bind ou des machines Windows 9x.

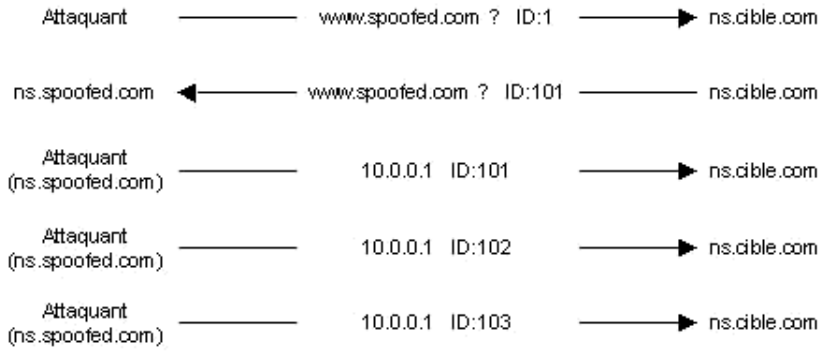
Dans tous les cas, il est nécessaire de répondre avant le serveur DNS, en le faisant tomber via un déni de service par exemple.

Pour parvenir à ses fins, l'attaquant doit contrôler un serveur DNS (`ns.attaquant.com`) ayant autorité sur le domaine `attaquant.com`. Le serveur DNS cible (`ns.cible.com`) est supposé avoir des numéros de séquence prévisibles (s'incrémentant de 1 à chaque nouvelle requête).

L'attaque se décompose en quatre étapes :

1. l'attaquant envoie une requête DNS pour le nom `www.attaquant.com` au serveur DNS du domaine `cible.com` comme le montre la figure 11;
   Fig.11: Envoi de la requête DNS à `ns.cible.com`

2. le serveur DNS cible a donc relayé la demande au DNS du domaine `attaquant.com`;
3. l'attaquant est capable de sniffer la requête pour récupérer son ID (dans notre exemple l'ID a une valeur de 100);
4. l'attaque falsifie l'adresse IP associée à un nom de machine, ici la machine victime est `www.spoofed.com` qui a normalement l'adresse IP `192.168.0.1`. Le pirate émet une requête DNS de résolution du nom `www.spoofed.com` vers `ns.cible.com`. Immédiatement après, il envoie une multitude de réponses DNS falsifiées (donnant comme adresse IP celle du site de l'attaquant `10.0.0.1`) à cette même requête en ayant spoofé l'adresse IP source avec celle du serveur DNS du domaine `spoofed.com`. L'ID de chaque réponse sera incrémenté de 1 à partir de celui récupéré lors de la deuxième étape (ID = 100) pour augmenter la chance de tomber sur le bon numéro d'ID réponse, dans le cas où `ns.cible.com` aurait du répondre à d'autres requêtes et donc incrémenté son ID DNS. La figure 12 détaille cette étape.
   Fig.12: DNS ID Spoofing



Le cache du serveur DNS cible est donc corrompu, et la prochaine machine demandant une résolution du nom `www.spoofed.com` récupérera l'adresse IP de la machine de l'attaquant et sera redirigée vers son site qui pourra être une copie du vrai site pour tromper les internautes et leur voler des informations confidentielles.

DNS Cache Poisoning

Les serveurs DNS possèdent un cache gardant en local, pendant un certain temps, les réponses de requêtes passées. Ceci pour éviter de perdre du temps à interroger chaque fois le serveur de nom ayant autorité sur le domaine demandé. Ce deuxième type de DNS Spoofing va consister à corrompre ce cache avec de fausses informations. Voici un exemple de cache poisoning :

Les conditions de l'exemple précédent sont toujours valables. Voici les différentes étapes de l'attaque :

- envoyer une requête DNS de résolution du nom `www.attaquant.com` au serveur DNS du domaine `cible.com`;
- Le serveur DNS cible envoie donc une requête portant sur une résolution du nom

`www.attaquant.com` au serveur DNS de l'attaquant;
- Le serveur DNS de l'attaquant enverra une réponse avec des enregistrements falsifiés qui permettront d'assigner un nom de machine avec une adresse IP appartenant à l'attaquant. Par exemple, le site `www.cible.com` pourra avoir un enregistrement DNS falsifié renvoyant l'adresse IP de `www.attaquant.com` au lieu de la bonne adresse IP.

Les attaques applicatives

Les attaques applicatives s'appuient principalement sur des vulnérabilités spécifiques aux applications utilisées. Cependant, certaines attaques peuvent être classées par type.

Les problèmes de configuration

Un des premiers problèmes de sécurité engendré par les applications est celui des erreurs de configurations. Nous distinguerons deux types d'erreurs : les installations par défaut et les mauvaises configurations à proprement parler.

Des logiciels, comme les serveurs Web, installés par défaut ont souvent des sites exemples qui peuvent être utilisés par des pirates pour accéder à des informations confidentielles. Par exemple, il peut y avoir des scripts permettant d'obtenir les sources des pages dynamiques ou des informations sur le système utilisé. En outre, lors d'une telle installation une interface d'administration à distance est disponible avec un login/mot de passe par défaut (trouvable dans le guide d'administration de l'application). Le pirate a donc la main sur le site et peut le modifier selon son bon vouloir.

Les principales failles générées par une mauvaise configuration sont des listes d'accès mal paramètrées. Le pirate accède alors à des pages et autres bases de données privées.

Comme exemple classique de problème de configuration, les erreurs de paramétrage du serveur Web Lotus Domino sont courantes. En effet, lors de l'installation de ce serveur, des bases Lotus de configuration sont accessibles sans aucune liste de contrôle d'accès. Concrètement, si la base Lotus *names.nsf* est accessible via un navigateur Web sans demande d'authentification, il est alors possible d'obtenir de nombreuses informations comme le nom de tous les utilisateurs Lotus. Cette base n'est qu'un exemple, et Lotus Domino en contient un nombre important de sensibles.

Les "bugs"

Une mauvaise programmation des logiciels entraîne obligatoirement des "bugs". Ceux-ci seront la source des failles de sécurité les plus importantes. Ces vulnérabilités quand elles sont découvertes vont permettre d'exécuter des commandes non autorisées, obtenir le source de pages dynamiques, rendre indisponible un service, prendre la main sur la machine, etc. Les plus connus de ces "bugs" et les plus intéressants en ce qui concerne leur exploitation sont les buffer overflow.

Les buffer overflow

Le dépassement de pile est une faille due à une mauvaise programmation. Effectivement, un buffer overflow apparaît quand une variable passée en argument d'une fonction est recopiée dans un buffer sans que sa taille n'aie été vérifiée. Il suffit que la variable ait une taille supérieure à l'espace mémoire réservé pour ce buffer pour qu'un dépassement de pile se produise. Celui-ci sera exploité en passant dans la variable un fragment de programme capable de faire démarrer un shell tout en obligeant ce débordement de pile à se produire. Dans le cas où un pirate réussi cette attaque il obtiendra alors le moyen d'exécuter à distance des commandes sur la machine cible avec les droits de l'application attaquée. Le buffer overflow et son exploitation ont été le sujet d'une série d'articles sur la programmation sécurisée :

- Eviter les failles de sécurité lors du développement d'une application - Partie 1
- Eviter les failles de sécurité lors du développement d'une application - Partie 2: mémoire, pile et fonctions, shellcode
- Eviter les failles de sécurité lors du développement d'une application - Partie 3: débordements de buffer
- Eviter les failles de sécurité lors du développement d'une application - Partie 4: chaînes de format
- Eviter les failles de sécurité lors du développement d'une application - Partie 5: race conditions
- Eviter les failles de sécurité lors du développement d'une application - Partie 6: scripts CGI

Les scripts

Une mauvaise programmation des scripts a souvent une répercution sur la sécurité d'un système. En effet, il existe des moyens d'exploiter des failles de scripts développés en Perl qui permettront de lire des fichiers hors racine Web ou d'exécuter des commandes non autorisées. Ces problèmes de programmation ont été évoqués dans l'un des articles ci-dessus, sur les failles des développements en Perl et PHP (Partie 6, les scripts CGI).

Man in the middle

L'objectif principal de cette attaque est de détourner le trafic entre deux machines. Cela pour intercepter, modifier ou détruire les données transmises au cours de la communication. Cette attaque est plus un concept qu'une attaque à part entière. Il existe plusieurs attaques mettant en oeuvre ce principe du Man in The Middle, comme le DNS Man in the Middle qui qu'une utilisation du DNS Spoofing pour détourner le trafic entre un client et un serveur Web. De même, une application récente a été élaborée pour détourner du trafic SSH.

Les dénis de service

Cette attaque porte bien son nom puisque qu'elle aboutira à l'indisponibilité du service (application spécifique) ou de la machine visée. Nous distinguerons deux types de déni de services, d'une part ceux dont l'origine est l'exploitation d'un bug d'une application et d'autre part ceux dus à une mauvaise implémentation d'un protocole ou à des faiblesses de celui-ci.

Les dénis de service applicatifs

Tout comme les vulnérabilités d'une application entraînent la possibilité de prendre le contrôle d'une machine (exemple du buffer overflow), elles peuvent aussi amener à un déni de service. L'application sera alors indisponible par saturation des ressources qui lui sont allouées ou un crash de celle-ci.

Les dénis de service réseaux

Il existe différents types de déni de service utilisant les spécificités des protocoles de la pile TCP/IP.

SYN Flooding

Nous avons vu qu'une connexion TCP s'établie en trois phases (TCP Three Way Handshake). Le SYN Flooding exploite ce mécanisme d'établissement en trois phases. Les trois étapes sont l'envoi d'un SYN, la réception d'un SYN-ACK et l'envoi d'un ACK. Le principe est de laisser sur la machine cible un nombre important de connexions TCP en attentes. Pour cela, le pirate envoie un très grand nombre de demandes de connexion (flag SYN à 1), la machine cible renvoie les SYN-ACK en réponse au SYN reçus. Le pirate ne répondra jamais avec un ACK, et donc pour chaque SYN reçu la cible aura une connexion TCP en attente. Etant donné que ces connexions semi-ouvertes consomment des ressources mémoires au bout d'un certain temps la machine est saturée et ne peut plus accepter de connexion. Ce type de déni de service n'affecte que la machine cible.

Le pirate utilise un SYN Flooder comme synk4, en indiquant le port TCP cible et l'utilisation d'adresses IP source aléatoires pour éviter toute identification de la machine du pirate.

UDP Flooding

Ce déni de service exploite le mode non connecté du protocole UDP. Il crée un "UDP Packet Storm" (génération d'une grande quantité de paquets UDP) soit à destination d'une machine soit entre deux machines. Une telle attaque entre deux machines entraîne une congestion du réseau ainsi qu'une saturation des ressources des deux hôtes victimes. La congestion est plus importante du fait que le trafic UDP est prioritaire sur le trafic TCP. En effet, le protocole TCP possède un mécanisme de contrôle de congestion, dans le cas où l'acquittement d'un paquet arrive après un long délai, ce mécanisme adapte la fréquence d'émission des paquets TCP, le débit diminue. Le protocole UDP ne possède pas ce

mécanisme, au bout d'un certain temps le trafic UDP occupe donc toute la bande passant n'en laissant qu'une infime partie au trafic TCP.

L'exemple le plus connu d'UDP Flooding est le *Chargen Denial of Service Attack*. La mise en pratique de cette attaque est simple, il suffit de faire communiquer le service `chargen` d'une machine avec le service `echo` d'une autre. Le service `chargen` génère des caractères tandis que `echo` se contente de réémettre les données qu'il reçoit. Il suffit alors au pirate d'envoyer des paquets UDP sur le port 19 (`chargen`) à une des victimes en spoofant l'adresse IP et le port source de l'autre. Dans ce cas le port source est le port UDP 7 (`echo`). L'UDP Flooding entraîne une saturation de la bande passante entres les deux machines. Un réseau complet peut donc être victime d'un UDP Flooding.

Packet Fragment

Les dénis de service de type Packet Fragment utilisent des faiblesses dans l'implémentation de certaines pile TCP/IP au niveau de la défragmentation IP (réassemblage des fragments IP).

Une attaque connue utilisant ce principe est Teardrop. L'offset de fragmentation du second fragment est inférieur à la taille du premier ainsi que l'offset plus la taille du second. Cela revient à dire que le deuxième fragment est contenu dans le premier (overlapping). Lors de la défragmentation certains systèmes ne gèrent pas cette exception et cela entraîne un déni de service. Il existe des variantes de cette attaque : bonk, boink et newtear par exemple. Le déni de service Ping of Death exploite une mauvaise gestion de la défragmentation au niveau ICMP, en envoyant une quantité de données supérieure à la taille maximum d'un paquet IP. Ces différents dénis de services aboutissent à un crash de la machine cible.

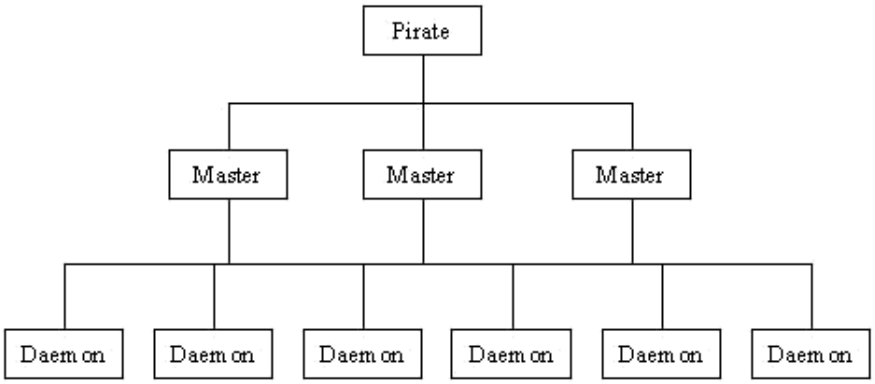
Smurfing

Cette attaque utilise le protocole ICMP. Quand un ping (message ICMP ECHO) est envoyé à une adresse de broadcast (par exemple 10.255.255.255), celui-ci est démultiplié et envoyé à chacune des machines du réseau. Le principe de l'attaque est de spoofer les paquets ICMP ECHO REQUEST envoyés en mettant comme adresse IP source celle de la cible. Le pirate envoie un flux continu de ping vers l'adresse de broadcast d'un réseau et toutes les machines répondent alors par un message ICMP ECHO REPLY en direction de la cible. Le flux est alors multiplié par le nombre d'hôte composant le réseau. Dans ce cas tout le réseau cible subira le déni de service, puisque l'énorme quantité de trafic généré par cette attaque entraîne une congestion du réseau.

Déni de service distribué

Les dénis de services distribués saturent le réseau victime. Le principe est d'utiliser plusieurs sources (daemons) pour l'attaque et des maîtres (masters) qui les contrôlent. Les outils de DDoS (*Distributed Denial of Service*) les plus connus sont Tribal Flood Network (TFN), TFN2K, Trinoo et Stacheldraht. La figure 13 montre un réseau typique de DDoS :

Pic.13: Réseau DDoS



Le pirate utilise des maîtres pour contrôler plus facilement les sources. En effet, il a besoin de se connecter (en TCP) aux maîtres pour configurer et préparer l'attaque. Les maîtres se contentent d'envoyer des commandes aux sources en UDP. S'il n'y avait pas les maîtres, le pirate serait obligé de se connecter à chaque source. La source de l'attaque serait détectée plus facilement et sa mise en place beaucoup plus longue.

Chaque daemon et master discutent en échangeant des messages spécifiques selon l'outil utilisé. Ces communications peuventt même être cryptées et/ou authentifiées. Pour installer les daemons et les masters, le pirate utilise des failles connues (buffer overflow sur des services RPC, FTP ou autres). L'attaque en elle-même est un SYN Flooding, UDP Flooding ou autre Smurf Attack. Le résultat d'un déni de service est donc de rendre un réseau inaccessible.

Conclusion

Aujourd'hui, la sécurité contre les attaques à distance se renforce de plus en plus contrairement à la sécurité interne. Ce parent pauvre de la protection contre les pirates laisse encore de belles perspectives à des attaques en local comme le TCP Session Hijacking, l'ARP Spoofing et le DNS Spoofing. Par ailleurs, la prédiction de numéro de séquence (coeur de l'IP Spoofing) et les variantes de Fragments Attacks apparaissent uniquement à cause de bugs au niveau des OS des équipements réseaux. En ce qui concerne les attaques applicatives, elles ont encore de beaux jours devant elles grâce à la complexité sans cesse croissante des applications liées au Web et aux délais de plus en plus courts imposés aux développeurs et aux administrateurs. Quant aux dénis de service, ils seront toujours aussi redoutables dans leur forme distribuée tant que tout le monde ne sera pas sensibilisé à la protection de leur machine personnelle.

Liens

- RFC 1858 - Security Considerations for IP Fragment Filtering : sunsite.dk/RFC/rfc/rfc1858.html

- IP Spoofing Demystified - Phrack 48 : www.phrack.org/
- Simple Active Attack Against TCP - Laurent Joncheray : www.insecure.org/stf/iphijack.txt
- DNS ID Hacking - ADM Crew : packetstorm.securify.com/groups/ADM/ADM-DNS-SPOOF/ADMID.txt
- The DoS Project’s "trinoo" - David Dittrich : staff.washington.edu/dittrich/misc/trinoo.analysis
- The Strange Tale of the DENIAL OF SERVICE Attacks against GRC.COM : grc.com
- hping2 : www.kyuzz.org/antirez/hping.html
- nemesis : www.packetfactory.net/Projects/nemesis/
- mendax : packetstorm.securify.com/Exploit_Code_Archive/mendax_linux.tgz
- hunt : lin.fsid.cvut.cz/~kra/index.html
- dsniff : www.monkey.org/~dugsong/dsniff/
- fragrouter : packetstorm.securify.com/UNIX/IDS/fragrouter-1.6.tar.gz

---

2005-01-14, generated by lfparser_pdf version 2.51