La protection des réseaux contre les attaques DOS

Dany Fernandes et *Papa Amadou Sarr*

Mai 2010

Tuteur : Osman SALEM

Sommaire

Remerciements

Avant d’entamer ce rapport, nous tenons à remercier le tuteur de notre projet, M. Osman Salem pour toute l’aide qu’il nous a apporté durant ces 5 mois de projets. Nous remercions également Dominique Seret pour l’intérêt porté à notre projet et également pour son cours d’SSIC du premier semestre qui nous a donné envie de poursuivre dans la sécurité informatique. Enfin, nous remercions Ahmed Mehaoua pour son cours de réseau avancé qui nous a permis d’avoir les bases de réseau nécessaire à la réalisation du projet.

Introduction

Aujourd'hui, les réseaux informatiques sont de plus en plus développés, que se soit chez les particuliers ou dans le domaine professionnel. L'expansion des systèmes informatiques mais surtout de l'internet ces dernières années ont rendu les réseaux indispensables pour les entreprises.

Mais si toutes ces innovations ont apportés de très nombreux avantages aux entreprises, elles sont accompagnées de nouveaux risques inhérents à ces nouvelles technologies, le piratage informatique. En effet, ces attaques sont de plus en plus nombreuses, efficaces et simple à mettre en œuvre. Elles sont utilisées pour l'espionnage industrielle (vols d'informations confidentielles) ou simplement du parasitage (destruction de données numériques ou arrêt de service). Dans le cadre de notre projet, nous nous intéresserons aux attaques de types dénis de service, qui sont les plus repandues. Ce type d'attaque n'est pas dangereux pour les données numériques du réseau, mais a pour objectif de rendre indisponible un service proposé par une entreprise, par exemple un site de commerce en ligne ; une entreprise ne peut se permettre d'avoir son site indisponible pendant plusieurs heures ou jours.

Dans un premier temps, nous allons définir ce qu'est le déni de service, pour comprendre leur fonctionnement et leur mise en œuvre. Nous verrons ensuite quels sont les moyens qui sont à notre disposition pour se prémunir de ce genre d'attaque.

1. Une attaque connue : le déni de service

1.1. Qu'est ce que le déni de service

Un « déni de service » (Denial of Service ou *DoS*) est une attaque réalisée dans le but de rendre indisponible durant une certaine période les services ou ressources d'une organisation. Généralement, ce type d'attaque à lieu contre des machines et serveurs d'une entreprise afin qu'ils deviennent inaccessibles pour leurs clients.

Le but d'une telle attaque n'est pas d'altérer ou de supprimer des données, ni même de voler quelconque information. Il s'agit ici de nuire au fonctionnement d'un service ou à la réputation d'une société qui offre un service sur Internet en empêchant le bon fonctionnement de celui ci.

Réaliser un déni de service est aujourd'hui simple, et également très efficace. Il est possible d'attaquer tout type de machine (Windows, Linux, Unix…) car la plupart des dénis de service exploitent des failles liées au protocole TCP/IP.

Il en existe principalement deux types :

1. Les dénis de service par saturation qui consistent à submerger une machine de requêtes, afin qu'elle ne soit plus capable de répondre aux requêtes réelles ;
2. Les dénis de service par exploitation des vulnérabilités qui consistent à exploiter une faille du système afin de le rendre inutilisable.

Le principe de ces attaques est d'envoyer des paquets ou des données de taille ou de constitution inhabituelle, afin de provoquer une saturation ou un état instable des machines victimes et de les empêcher ainsi d'assurer les services réseau qu'elles sont sensées offrir. Dans certains cas extrêmes, ce type d'attaque peut conduire au crash de la machine cible.

Le déni de service est donc un type d'attaque qui coûte très cher puisqu'il interrompt le cours normal des transactions d'une organisation. Sachant qu'à l'heure actuelle, les sommes et les enjeux d'une entreprise sont généralement énormes, cela peut poser de graves problèmes si une telle situation se produit ne fût-ce que quelques minutes.

Les contre-mesures sont compliquées à mettre en place et doivent être spécifiques à un type d'attaque. Etant donné que les attaques par déni de service utilisent les services et protocoles normaux d'Internet, s'en protéger reviendrait à couper les voies de communications normales, sachant qu'il s'agit de la raison d'être principale des machines concernées (serveurs web, serveur mail ...).

Il faut donc essayer se protéger au mieux de certains comportements anormaux, ce qui implique notamment la vérification de l'intégrité des paquets, la surveillance du trafic, établissement de profils types et de seuils, etc. On est donc loin de la protection absolue, mais il est tout de même possible de se protéger de façon intelligente et flexible.

1.2. Déni de service distribué

Lorsqu'un déni de service est provoqué par plusieurs machines, on parle alors de « déni de service distribué » (Distributed Denial of Service, ou *DDoS*). Le but recherché du déni de service sont les mêmes que pour le DoS, à la différence près que plusieurs machines à la fois sont à l'origine de l'attaque (c'est une attaque distribuée).

L'attaquant va donc se constituer un réseau où chacune des machines va attaquer la cible à un moment donné. Ce réseau se compose d'un maître et de nombreux hôtes distants, encore appelés démons. Pendant le déroulement de l'attaque, le hacker se connecte au maître qui envoie alors un

ordre à tous les hôtes distants. Ensuite, ceux-ci vont attaquer la cible suivant une technique choisie par le hacker. Il existe des attaques de type « agressives », dont le but de faire crasher complètement la cible, ou encore des attaques de type "stream" (TCP ACK sur des ports au hasard).

Les DDoS se sont démocratisées depuis quelques années. En effet, à leur début, ces attaques nécessitaient de bonnes connaissances de la part des attaquants. Mais à présents, il existe des outils pour organiser et mettre en place l'attaque. Ainsi le processus de recherche des hôtes secondaires (ou zombies) a été automatisé. En repérant certaines failles courantes sur les machines présentes sur Internet, l'attaquant finit par se rendre maître (accès administrateur) de centaines voir de milliers de machines non protégées. Il installe ensuite les clients pour l'attaque secondaire et essaye également d'effacer ses traces. Une fois le réseau en place, il n'y a plus qu'à donner l'ordre pour inonder la victime finale de paquets inutiles.

Il est intéressant de noter que les victimes dans ce type d'attaques ne sont pas que celles qui subissent le déni de service; tous les hôtes secondaires sont également des machines compromises jusqu'au plus haut niveau (accès root), tout comme l'hôte maître. La menace provient du fait que les outils automatisant le processus ont été très largement diffusés sur Internet. Il n'y a plus besoin d'avoir des connaissances pointues pour la mettre en place.

Ce type d'attaque reste très difficile à contrer ou à éviter : il s'agit donc d'une menace que beaucoup craignent. En effet, cette attaque est très dévastatrice, et ne provient plus seulement d'une seule machine, mais d'un réseau tout entier. Sachant le nombre de machines non sécurisées présentes sur Internet, on peut imaginer l'ampleur d'une telle attaque.

Il n'est donc pas évident de s'en protéger étant donné que l'identité des attaquants change souvent et que le temps nécessaire pour organiser une protection adéquate est bien souvent supérieur au temps nécessaire pour mettre à mal la victime. Il est donc avant tout primordial de localiser l'initiateur et de repérer sa signature. La détection d'un trafic suspect peut servir de prévention.

1.3. Les principaux types d'attaque

Il existe de nombreux types d'attaques par déni de service. Le simple fait d'empêcher un système de rendre un service peut être qualifié de « déni de service ». En voici les principaux :

- Ping flooding : cette attaque consiste à envoyer un flux maximal de « ping » vers une cible.
- Attaque Ping of Death : le principe est d'envoyer un paquet ICMP avec une quantité de données supérieure à la taille maximale d'un paquet IP. Si la cible n'est pas adaptée à gérer ce type de paquet, il peut se produire un crash.
- SYN flood : cette technique consiste à saturer une machine en envoyant une multitude de paquets TCP avec le flag SYN armé. Cela créera une multitude de connexions en attente, demandant un grand nombre des ressources du système.
- UDP Flood : l'attaquant envoie un grand nombre de requêtes UDP sur une machine. Le trafic UDP étant prioritaire sur le trafic TCP, ce type d'attaque peut vite troubler et saturer le trafic transitant sur le réseau.
- L'attaque ARP : elle consiste à s'attribuer l'adresse IP de la machine cible, c'est-à-dire à faire correspondre son adresse IP à l'adresse MAC de la machine pirate dans les tables ARP des machines du réseau. Pour cela il suffit en fait d'envoyer régulièrement des trames ARP REPLY en diffusion, contenant l'adresse IP cible et la fausse adresse MAC. Cela a pour effet de modifier les tables dynamiques de toutes les machines du réseau. Celles-ci enverront donc leurs trames Ethernet à la machine pirate tout en croyant communiquer avec la cible
- L'attaque Teardrop : cette attaque utilise une faille propre à certaines piles TCP/IP. Cette vulnérabilité concerne la gestion de la fragmentation IP. Ce problème apparaît lorsque la pile reçoit le deuxième fragment d'un paquet TCP contenant comme donnée le premier fragment.

La pile TCP/IP peut s'avérer incapable de gérer cette exception et le reste du trafic.

- Smurfing : consiste à envoyer un ping en diffusion sur un réseau (A) avec une adresse IP source correspondant à celle de la cible (B). Le flux entre le port ping de la cible (B) et du réseau (A) sera multiplié par le nombre de machines sur le réseau (A), conduisant à une saturation de la bande passante du réseau (A) et du système de traitement des paquets de (B).
- L'attaque land : consiste à envoyer des paquets TCP comportant une adresse source IP et un numéro de port identiques à ceux de la victime. Le host attaqué pense alors qu'il parle à lui même ce qui généralement provoque un crash.
- Les bombes e-mail : le mail bombing consiste à envoyer de gros ou de nombreux fichiers à un utilisateur pour saturer sa boîte de réception de courrier électronique.
- L'attaque Unreachable Host : cette attaque envoie des messages ICMP "Host Unreachable" à une cible, provoquant la déconnexion des sessions et la paralysie de la victime, même si ils sont envoyés à faible cadence.

1.4. Historique des grandes attaques

- Nuke en avril 1992 (Icmp_unreach)
- Octopus en janvier 1996 (While – connect)
- Ping Of Death en décembre 1996
- Smurf en juillet 1997
- Land en octobre 1997 (Windows 95, NT 4.O et 98)
- Latierra en octobre 1997 (Plante Windows 95 et occupe 100% du CPU sur NT 4.0)
- Teardrop / Overdrop en décembre 1997 (Plante Linux, NT et 95)
- Syndrop en juin 1998 (Teardrop en tcp avec le bit syn et des champs invalides)
- Snork en septembre 1998 (Tueur de Windows NT, envoi de paquet RPC (135/UDP) de la part d'un autre NT)
- Smack en octobre 1998 (Inondation de paquets ICMP-UNREACHABLE aléatoires)
- Attaque sur le serveur de mise à jour de Microsoft
- Attaque de sites Web connus tels que Google, Microsoft, Apple Computer …
- Attaques de type « ping flood » en octobre 2002 sur les serveurs racines DNS…

1.5. La nouvelle arme des pirates

Au delà du fait qu'un déni de service occasionne beaucoup de problèmes, on voit se profiler de plus en plus une nouvelle technique de chantage que bons nombres de pirates exercent maintenant allégrement. En effet, les sociétés dont la principale activité est basée sur un flux d'information sur l'Internet (souvent avec des sommes importantes en jeux) sont menacées, et susceptibles d'être paralysées à tout moment, occasionnant des pertes importantes, et cela, à tout niveau.

Les pirates actuels utilisent donc les attaques par déni de service comme nouvelle arme pour exercer un chantage contre les sociétés. Le message est clair : une rançon est demandée en échange de la « non-paralysie » de leur activité. Il est évident que le ralentissement, voir même le blocage de leurs services pendant quelques minutes pourrait occasionner de grandes pertes d'argent, ainsi que beaucoup de désagréments pour leurs clients. Ces sociétés ont donc tout intérêt à obéir ou à trouver une parade pour s'en protéger, sans quoi les menaces seraient mises à exécution.

C'est le cas de la société « DoubleClick », qui en fit les frais le 28 juillet 2004. Son site s'est retrouvé fortement ralenti - voire paralysé - pendant de longues heures, entraînant dans son infortune les sites de ses clients, qui dépendent de sa technologie pour gérer leurs bannières de publicité.

Ce cyber crime fut perpétré par un groupe de trois pirates russes, jusqu'à ce que la police locale ne les arrête. Leurs cibles étaient des banques et des sociétés de paris britanniques (bookmakers) que les pirates soumettaient à des dénis de service distribués et à des extorsions de fonds en échange de leur arrêt, soit entre 15 000 et 45 000 euros. Les attaques avaient lieu lors des pics d'audience des sites de paris, perturbant fortement leurs activités.

Cet exemple ne fait qu'illustrer le fait que cette nouvelle arme est maintenant utilisée de plus en plus couramment. Toutes les sociétés gérant de l'argent sont menacées, aucune n'est à l'abri. Toute société accessible par Internet est ouverte est potentiellement en danger. On remarque d'ailleurs le nombre d'attaques de ce genre augmente chaque année.

2. Le Syn Flooding

Parmi les attaques précédemment citées, nous nous sommes principalement focaliser sur les attaques de type Syn Flood pour saturer un service en l'inondant de requêtes. Sa mise en œuvre a requis l'utilisation d'attaques sous-jacentes comme l'IP-Spoofing, PING-Flood.

2.1. Nature de l'attaque

Une connexion TCP s'établit en trois phases (TCP Three Way Handshake). Le SYN Flooding exploite ce mécanisme d'établissement. Les trois étapes sont l'envoi d'un SYN, la réception d'un SYN-ACK et l'envoi d'un ACK.

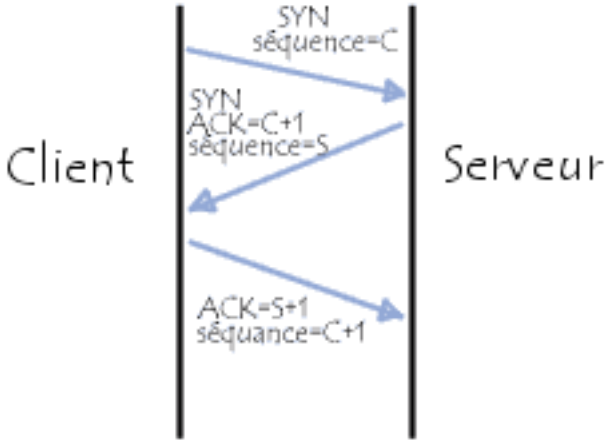


Figure 1 : Ouverture d'une connexion en TCP (extraite de : www.commentcamarche.net).

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 alors 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 dans une file d'attente ou queue de message. 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 nouvelle connexion. Ce type de déni de service n'affecte que la machine cible.

Le pirate emploie généralement l'IP Spoofing (création de paquet IP avec une adresse source falsifiée) afin de masquer son identité. Cependant, puisqu'il usurpe l'identité d'une autre machine, il lui faudra s'assurer que celle-ci ne répondra pas aux SYN-ACK émis par la victime (on entend par "réponse" une requête de type RESET précisant que le SYN-ACK reçu n'était pas attendu).

Les machines vulnérables aux attaques SYN mettent en file d'attente, dans une structure de données en mémoire, les connexions ainsi ouvertes, et attendent de recevoir un paquet ACK. Il existe un

mécanisme d'expiration permettant de rejeter les paquets au bout d'un certain délai. Néanmoins, avec un nombre de paquets SYN très important, si les ressources utilisées par la machine cible pour stocker les requêtes en attente sont épuisées, elle entre dans un état où elle ne peut fournir le service.

2.2. Réalisation de l'attaque

Nous avons mis en œuvre une attaque SYN FLOODING par le biais d'un serveur web apache. Ce dernier présente l'avantage de fonctionner sous différentes plates-formes telles que Windows et Linux, ce qui nous a permis de réaliser des tests variés.

Lors du premier test, nous avons utilisé apache sur une machine Linux. Le but était de rendre ce serveur indisponible lors de la demande de page web par des clients. Pour ce faire, nous avons utilisé l'outil HPING, un utilitaire gratuit (http://www.hping.org) qui permet de remplacer la commande ping en lui rajoutant plusieurs fonctionnalités et notamment l'envoi de requêtes TCP ou UDP suivant différents paramètres :

- intervalle entre les paquets,
- adresse source,
- adresse destination,
- taille des paquets,
- les flags,
- etc.

Bien utilisé, HPING permet d'effectuer un déni de service avec par exemple la commande suivante:

```
hping3 -i u10 192.168.0.11 -p 80 -S
```

qui envoi à l'adresse 192.168.0.11, sur le port 80, des paquets de type SYN à l'intervalle de 10 micro-secondes.

Mais avant tout, pour réaliser une attaque, la première étape consiste à scanner les ports ouverts d'une machine donnée pour en chercher les vulnérabilités. Pour cela, nous avons utilisé l'outil nmap:

```
nmap -p 1-100  192.168.0.11
```

Cette commande effectue le scan des ports 1 à 100 de la machine 192.168.0.11.

Comme nous l'avons dit précédemment, un pirate masque généralement son identité en falsifiant l'adresse IP source des paquets qu'il envoie. Egalement il doit s'assurer qu'il n'y aura pas de réponse de la part de la machine à qui on a emprunté l'adresse IP.

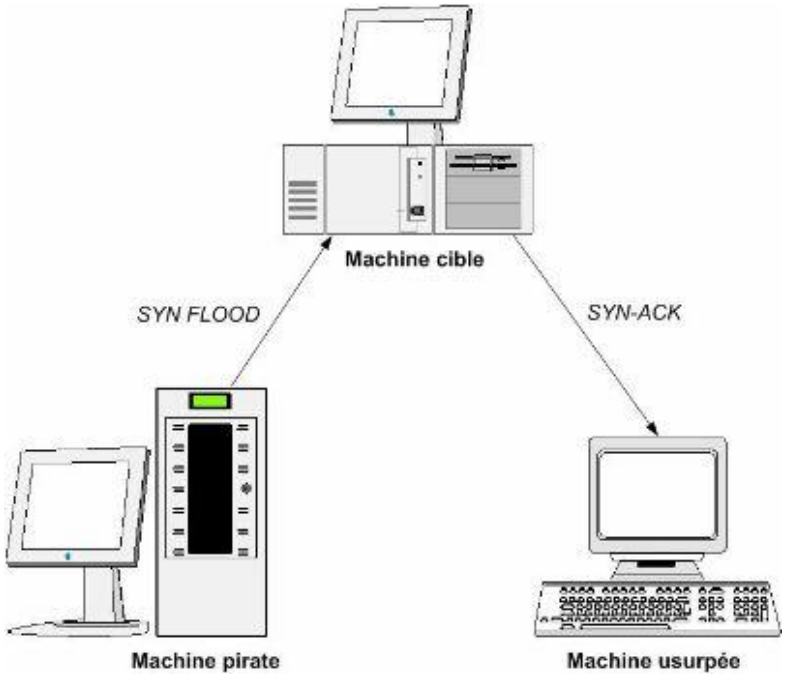


Figure 2 : Attaque SYN Flood (extraite de http://www.student.montefiore.ulg.ac.be/*)*.

Il a donc le choix d'usurper l'identité d'une machine existante, auquel cas il doit l'empêcher de répondre ; ou d'employer une adresse qui n'est pas attribuée, il n'y aura donc aucune machine pour répondre à la victime.

Si une machine dont l'IP a été usurpée venait à recevoir les SYN-ACK provenant de la victime, c'est à dire des paquets inattendus, elle y répondrait par un RST, ce qui mettrait fin à la connexion en attente et libérerait les ressources (empêchant donc l'attaque de se réaliser correctement).

Pour s'en assurer, le pirate doit donc inonder la machine cible avec une énorme quantité de SYN, tout en sachant que la machine usurpée était toujours active sur le réseau ; nous avons pu constater que quelque soit le nombre de SYN envoyé, la cible parvenait toujours à traiter les RST reçus et donc à libérer les ressources, ce qui prouve en effet qu'il faut réduire au silence la machine usurpée pour qu'elle n'envoie aucun RST.

Mise en pratique :

Notre environnement de travail se composait de 3 machines du réseau local 192.168.0.10 :

- 192.168.0.10 : la machine attaquante,
- 192.168.0.11 : la machine cible de l'attaque et son serveur web apache,
- 192.168.0.12 : une machine dont on usurpe l'identité,

Avec l'utilitaire HPING, nous pouvons créer des paquets avec le flag SYN positionné et les envoyer très rapidement grâce à la commande suivante :

```
hping –S –i u10 –p 80 –a 192.168.0.12 192.168.0.11
```

L'argument 'S' demande le positionnement du flag. Le second argument 'i' précise l'intervalle entre deux envois (ici en micro secondes). Le troisième 'p' spécifie le port destination. Le quatrième 'a' désigne une adresse source. Le dernier, enfin, donne l'adresse destination.

En pratique, pour une attaque «grandeur nature», il faut éviter que la machine dont nous avons usurpé l'identité ne puisse réponde à la machine attaquée. Mais dans le cadre de notre projet, nous

avons fait une attaque dans un réseau local, il nous a donc suffit de déconnecter la machine dont nous avons usurpé l'identité pour l'empêcher de répondre à la machine attaquée, et donc l'empêcher de libérer de l'espace mémoire.

2.3. Résultats de l’attaque

Nous avons pu constater que, lorsque l’attaque est bien effectuée, le serveur Web est complètement hors service, il est impossible d’obtenir une page provenant de celui-ci. L’attaque demande cependant un certain temps avant de consommer toutes les ressources du système (durant nos tests, cette période à été estimée entre 5 et 20 secondes suivant la configuration de la machine). Cette attaque fonctionne aussi bien sur Linux que sur Windows.

Il est nécessaire de se rendre compte de la simplicité de mise en œuvre de cette attaque. Un utilitaire disponible gratuitement et assez répandu et simple d'utilisation permet de mettre hors service un serveur non protégé en seulement quelques secondes.

2.4. Mesures de protection

2.4.1. Paramètres réseaux du noyau linux

Le noyau utilise de nombreux paramètres qui peuvent être ajustés en différentes circonstances. Bien que les paramètres par défaut puissent convenir, il peut être avantageux d'en modifier certains pour contrer, ou tout du moins limiter l'impact d'une attaque DOS sur une machine.

Une première mesure pour diminuer l'impact d'une attaque DOS est de modifier certains paramètres propres aux connexions TCP: le ***tcp_max_syn_backlog*** situé dans le fichier :

/proc/sys/net/ipv4/tcp_max_syn_backlog

Le tcp_max_syn_backlog correspond au nombre maximum de requêtes d'une connexion mémorisée, qui n'avait pas encore reçu d'accusé de réception du client connecté. La valeur par défaut est de 1024 pour des systèmes avec plus de 128 Mo de mémoire et 128 pour des machines avec moins de mémoire. Si un serveur souffre de surcharge, on peut essayer d'augmenter ce nombre.
L'autre paramètre modifiable est le ***tcp_keepalive_intvl*** qui se trouve dans le fichier:

/proc/sys/net/ipv4/tcp_keepalive_probes

Ce nombre correspond au temps d'attente avant retransmission d'une requête qui n'a pas été acquitté. Il est de 75 secondes par défaut. On peut le paramétrer à 30 secondes par exemple.

2.4.2. SYN-cookies

Le problème dans l'attaque de *SYN FLOODING* est le remplissage de la file d'attente des connexions à moitiés ouvertes. A chaque fois qu'un paquet SYN est reçu les informations concernant la connexion sont stockées dans une file. Une fois que l'attaquant à envoyé un nombre suffisant de paquets SYN la file se rempli complètement et de nouvelles connexions ne sont plus possibles, ce qui engendre l'indisponibilité du service. Il faudrait donc supprimer cette file d'attente. Le serveur n'a en fait pas besoin de stocker les informations de la connexion, contenues dans le paquet SYN (adresse IP et port du client et du serveur), vu qu'elles sont également présentes dans le paquet ACK envoyé par le client.

Un paquet ACK doit avoir les caractéristiques du paquet SYN/ACK envoyé par le serveur après une demande de connexion. Ces informations sont les suivantes : adresse IP et port du client, adresse IP et port du serveur, numéro de séquence du client, numéro de séquence du serveur. Ce dernier est choisi par le serveur. Le but est de choisir ce numéro en fonction des autres informations qui restent

identiques pour le paquet SYN initial et le paquet ACK du client. Il suffit ensuite de ne traiter les paquets ACK dont le numéro de séquence du serveur correspond à une fonction des autres informations présentes dans ce paquet (adresses IP, ports). On a donc inséré des informations dans le numéro de séquence du serveur, il n'y a donc plus besoin de stocker les informations de la connexion dans une file d'attente.

Le numéro de séquence est calculé de la manière suivante :

- les 5 premiers bits valent *t mod 32* où t est un compteur temporel allant de paire avec un secret.
- les 3 bits suivant correspondent au MSS (Maximum Segment Size) sélectionné par le serveur. Cet encodage est indispensable puisque le serveur ne conserve pas l'état de la connexion en cours de complétion et oublie donc plusieurs options TCP comme le MSS, le window scale etc.
- les derniers 24 bits correspondent au secret sélectionné par le serveur.

Ce secret est un hash MD5 de l'adresse source, de l'adresse de destination, du port source, du port de destination et enfin du compteur t.

Les SYN-cookies ne sont en fait activé que lorsqu'un flood est détecté, c'est-à-dire lorsque la file d'attente des paquets SYN est pleine. Avant cela les demandes de connexions sont traitées de manière tout à fait classique. Grâce à cette technique le service inondé de paquets SYN continuera à répondre aux demandes de connexions licites.

Les SYN-cookies sont implémentés dans le noyau Linux ainsi que dans FreeBSD mais pas activés par défaut. Ils ne sont pas présents sous Windows mais il existe d'autres façons de se protéger de cette attaque sur ce système.

Malheureusement les SYN-cookies ne sont pas compatibles avec certaines options TCP, notamment celles spécifiées dans les paquets SYN d'établissement de connexion, vu qu'ils ne conservent aucun état avant la complétion effective de la connexion. Par exemple, un encodage du MSS dans le paquet SYN à été adapté en utilisant 2 bits de numéro de séquence pour représenter 4 valeurs MSS communes prédéfinies. Il ne peut pas non plus retransmettre de paquet SYN/ACK en cas de disparité d'états entre le client et le serveur dû à des pertes par exemple. C'est principalement pour cette raison qu'ils ne sont pas activés par défaut. Ils ne devraient l'être que sur des serveurs susceptibles d'être victime d'une attaque de SYN flooding.

Un autre inconvénient des SYN-cookies est qu'ils utilisent un peu plus le CPU qu'une connexion ne les utilisant pas, vu qu'ils doivent générer un numéro de séquence en utilisant certaines informations du paquet SYN (adresses IP, ports). Mais cela reste raisonnable, ce n'est qu'une petite contrainte.

Enfin, étant donné que l'authentification de la connexion est basée uniquement sur la valeur du champ d'acquittement du paquet ACK envoyé par le client, si un attaquant arrive à deviner le protocole d'authentification utilisé par le serveur, il peut se faire passer pour n'importe qui. De plus il peut également inonder de paquets ACK le serveur pour tenter de créer une connexion. Et enfin, un serveur utilisant des **syncookies** ne pourra pas effectuer un filtrage de paquets ACK, ce qui peut permettre éventuellement à un attaquant de passer à travers un pare-feu qui filtre les paquets SYN de demande de connexion.

Pour activer les SYN-cookies :

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Pour les désactiver :

```
echo 0 > /proc/sys/net/ipv4/tcp_syncookies
```

2.4.3. Firewall

Les firewalls sont des équipements réseaux qui permettent de filtrer les paquets entrants et sortants afin de prévenir toutes attaques de l'extérieur. Ils se basent sur un fonctionnement séquentiel et un ensemble de règles pour autoriser seulement les connexions légitimes. Dans le cadre des DoS, le problème majeur est que les attaquant utilisent des connexions légitimes pour perpétrer leurs attaques. De plus, les firewalls ne peuvent pas efficacement différencier les connexions légitimes et illégitimes. Par contre ils peuvent se révéler très efficace pour contrer un attaquant. En se basant sur les informations fournis par des équipements de détections, on peut appliquer des règles très précises qui bloqueront uniquement les connexions malfaisantes, en se basant sur le protocole, l'IP ou le port.
De nombreux firewalls hardware ou software permettent de se prémunir contre les attaquants. Parmi eux, un des plus courants est le firewall intégré au noyau Linux : Netfilter et son interface iptables. Il présente l'avantage d'être open source donc gratuit et d'être assez facile à appréhender. De plus, cela n'a aucune influence sur sa puissance et sa modularité.
Quelques exemples de règles simples pour bloquer certaines attaques :
TPC syn flood:

```
iptables -A INPUT -p tcp --syn -m limit --limit 1/second -j ACCEPT
```

UDP flood:

```
iptables -A INPUT -p udp -m limit --limit 1/second -j ACCEPT
```

PING flood:

```
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/second -j ACCEPT
```

Bien maîtriser un firewall peut être une très bonne protection contre la majorité des attaques et des attaquants. Il est nécessaire de surveiller les connexions qui transitent sur son réseau pour être capable de bien se protéger contre toutes menaces.

2.4.4 Des moyens de préventions

2.4.4.1 La détection

Nous avons vu précédemment qu'il est très facile de mettre en place une attaque de type déni de service qui soit efficace. Pour se prémunir de ces attaques, on doit pouvoir être capable de détecter de manière efficace une attaque. Cependant, il peut être difficile d'identifier un paquet licite d'un paquet provenant d'un attaquant. Mais il existe plusieurs outils qui permettent avec plus ou moins d'efficacité de détecter/bloquer une attaque.

2.4.4.2 Les IDS

Un IDS (Intrusion Detection System) est un outil ou un ensemble d'outil dont l'objectif est de surveiller le trafic entrant et sortant du réseau, dans le but de détecter une attaque ou une intrusion dans le système et déclencher différentes alertes en fonction de sa configuration. Un IDS analyse le réseau en temps réel, il nécessite donc des ressources matériels mais aussi en bande passante.
Il existe deux types d'IDS:
Les NIDS (Network Based IDS) assure la sécurité au niveau réseau. Il va donc écouter tout le trafic du réseau et générer des alertes en cas de comportement anormal. Il se peut que l'on place plusieurs NIDS dans le réseau comme le montre le schéma suivant:

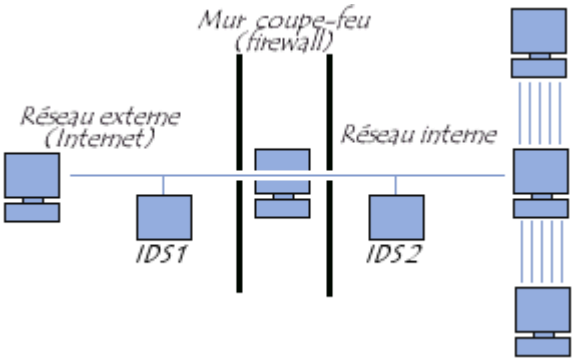


Figure 3 : NIDS (extraite de : www.commentcamarche.net).

Le premier IDS analyse tout le réseau entrant pour détecter toute tentative d'attaque, et la seconde analyse les requêtes qui ont franchis le premier IDS et le pare-feu, ou bien les requêtes internes au réseau.
Un NIDS nécessite donc beaucoup de ressources, en CPU notamment, ainsi qu'une force bande passante capable de supporter l'ensemble du trafic du réseau.

Le HIDS (Host Based IDS) réside sur une machine en particulier et non sur tout un réseau. Il est ainsi considéré comme un simple service, ou un démon d'un système. Le HIDS analyse le trafic de la machine hôte pour déceler des intrusions ou des attaques (déni de service par exemple). A la différence de l'NIDS, l'HIDS nécessite moins de ressources. Cependant, l'HIDS se basant sur l'état du système à l'installation, il faut donc que ce système soit sain pour prévenir tout risque d'intrusion. Un autre inconvénient, si l'on doit installer plusieurs machines, il faudra installer plusieurs HIDS. En revanche, un HIDS détecte peu de faux positifs.
Les IDS fonctionnent en plusieurs temps:

1ère étape: la capture du trafic.
Pour capturer le trafic, les IDS utilise la librairie libcap qui rend compatible l'IDS à toutes les plates-formes. L'IDS copie tout le trafic entrant puis utilise un filtre (par exemple Berckley Packet Filter) pour récupérer les informations utiles à l'analyse de celui-ci.

2ième étape: l'analyse.
Nous avons vu précédemment qu'il existe de nombreux types d'attaques, l'IDS procède donc à une analyse des scénarios d'attaques possibles. Tout comme un antivirus qui utilise une base de signature de virus pour les détecter, l'IDS une base de signature d'attaques ou règles qui vont définir le comportement à avoir dans le cas d'un scénario donné. Tout comme l'antivirus, il faut donc que la base de signatures d'attaques soit mise à jour régulièrement pour pouvoir contrer les nouveaux types d'attaques. L'analyse des scénarios permet de déceler les paquets illicites, par exemple dans le cas d'une attaque de type Ping-Of-Death où la taille du paquet ICMP n'a pas la taille conforme à sa définition dans le RFC. On peut également analyser le nombre de paquets entrant dans un port donné; s'il dépasse un certain seuil donné (notablement dans le cadre d'une attaque SYN-FLOOD), alors l'IDS peut lancer une alerte.

3ième étape: l'alerte.
Après avoir décelé un comportement illicite, l'IDS génère une alerte qui est généralement stockée dans un fichier de journalisation. Il peut également informer l'administrateur par mail qu'il a détectée un comportement suspect.

Mais aujourd'hui, les IDS ont tendance à laisser leur place aux IPS (Intrusion Prevention System).

L’IPS est un Système de protection contre les intrusions et non plus seulement de reconnaissance et de signalisation des intrusions comme la plupart des IDS le sont.

2.4.4.3 Les IPS

Le fonctionnement d'un IPS est similaire à celui d'un IDS. Il capture le trafic du réseau puis l'analyse. Mais au lieu d'alerter l'utilisateur d'une intrusion ou d'une attaque, l'IPS bloque directement les intrusions en supprimant les paquets illégitimes. Pour informer l'utilisateur, l'IPS peut aussi remplir un fichier de journalisation qui contiendra la liste des paquets supprimés et éventuellement un message indiquant la raison de cette suppression.
Cet outil est très efficace pour contrer les attaques ou intrusions extérieurs mais possèdes quelques inconvénient:

- Puisque l'IPS bloque ou supprime directement les paquets qu'il considère illégitimes sans en alerter l'administrateur, il se peut que des faux positifs soient rejetés par erreur, notamment si le service est utilisé très fortement, l'IPS peut considérer qu'il subit un déni de service si ces règles sont mal définies.
- Un autre inconvénient réside dans le fait que, dans le cadre d'un NIPS, si l'attaquant arrive à spoofer l'adresse IP d'un équipement du réseau, alors l'IPS bloquera cet équipement lorsqu'il détectera l'attaque. Mais il existe maintenant des techniques qui permettent d'éviter ce genre de problème, en écrivant une liste des adresses IP à ne pas supprimer.
- Dernier inconvénient, lorsqu'une attaque et lancée et arrêtée par l'IPS, celui-ci est détecté par l'attaquant qui va essayer de contourner l'IPS.

C'est pourquoi les IDS sont plus utilisés que les IPS, bien que certains IDS ont des fonctionnalités qui permettent de bloquer le trafic illégitime comme un IPS. C'est notamment le cas de Snort, l'IDS que nous avons utilisés dans le cadre de notre projet.

2.4.4.4 Snort

Définition de snort:

Snort est un système de détection d'intrusion libre sous licence GPL. À l'origine écrit par Martin Roesch, il appartient actuellement à Sourcefire (dont l'acquisition par Check Point avait été prévue en 2005 mais a été annulée par la suite). Des versions commerciales intégrant du matériel et des services de supports sont vendues par Sourcefire.

Snort est capable d'effectuer aussi en temps réel des analyses de trafic et de logger les paquets sur un réseau IP. Il peut effectuer des analyses de protocole, recherche/correspondance de contenu et peut être utilisé pour détecter une grande variété d'attaques et de sondes comme des dépassements de buffers, scans, attaques sur des CGI, sondes SMB, essai d'OS fingerprintings et bien plus. Cependant, comme tout logiciel, Snort n'est pas infaillible et demande une mise à jour régulière.

Snort peut également être utilisé avec d'autres projets open sources tels que SnortSnarf, ACID, sguil et BASE (Basic Analysis and Security Engine) afin de fournir une représentation visuelle des données concernant les éventuelles intrusions.

Source: http://fr.wikipedia.org/wiki/Snort

Fonctionnement de Snort:

Snort est composé de 4 blocs:

1. Le sniffer de paquets et le décodeur se basent sur la librairie libpcap pour lire les trames qui circulent sur le réseau
2. Le préprocesseur peut ensuite repérer les malformations, anomalies, etc. et les réparer.

3. Le moteur de détection se base sur les flux normalisés et ré-assemblés pour repérer d'éventuelles
4. Enfin, les événements sont inscrits (logs au format unifié, base de données, etc.).

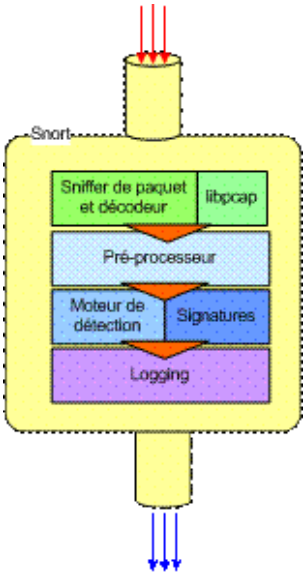


Figure 4 : fonctionnement de Snort.

A l'origine, l'IDS Snort avait une version modifiée appelée snort_inline qui transformait l'IDS en IPS. Mais aujourd'hui, les deux projets ont été fusionnés et à l'installation de Snort, on peut lui ajouter des options pour qu'il se comporte comme snort_inline, c'est-à-dire comme un IPS. Voici un schéma récapitulatif du fonctionnement de snort_inline (devenu Snort aujourd'hui).

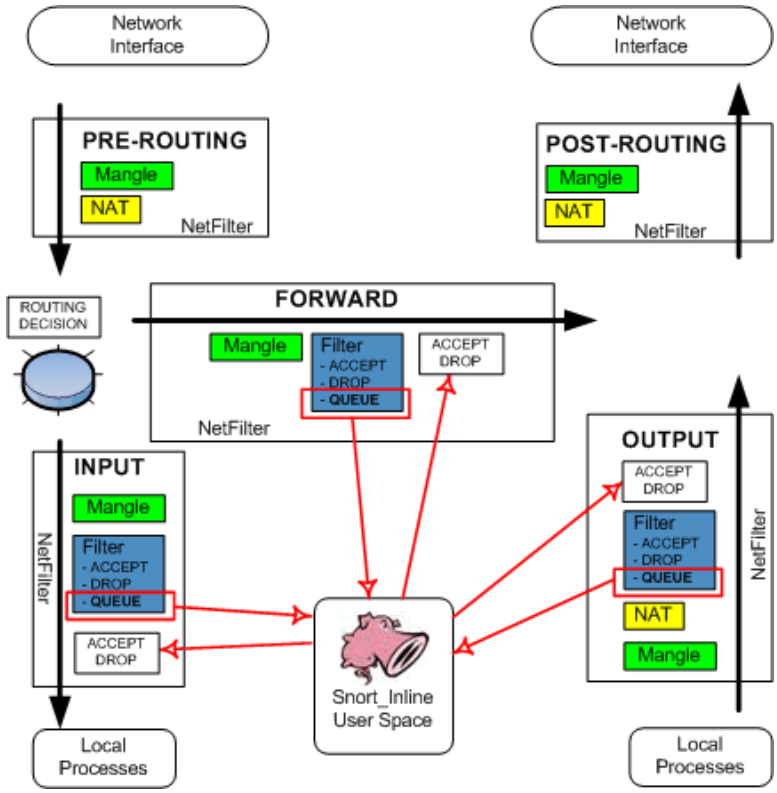


Figure 4 : Interaction entre Snort et iptables (http://openmaniak.com)

NetFilter est un module du noyau de Linux disponible depuis la version 2.4 du noyau. Il fournit trois principales fonctionnalités:

- Filter : filtrage de paquet (Accepter ou rejetter des paquets)

-NAT : change la source ou la destination IP d'un paquet réseau
-Packet Mangling : Modifie les paquets (utilisé par exemple pour la qualité de service, QoS) Iptables est un outil permettant de configurer NetFilter.

Netfilter met dans une queue de message les paquets vers Snort_Inline qui n'ont pas été rejetés par iptables, dans l'espace utilisateur (user space) avec l'aide du module du noyau Linux ip_queue et libipq. Puis, si un paquet correspond à une signature d'attaque de Snort_Inline, il est marqué par libipq et revient vers le noyau où est rejeté.

2.5. Détection de l'attaque

2.5.1. Snort et la règle de détection

Pour débuter, nous avons examiné la règle présente par défaut dans Snort pour les attaques de Syn Flood. Malheureusement celle-ci était sans intérêt dans le cadre de notre attaque. Nous avons alors créé notre propre règle de base, que voici :

drop tcp any any -> 192.168.1.17 80 (msg : "Syn flood sucks but our rule rules!"; flow: stateless; flags: S; sid:1000001;)

Notre règle permet à Snort de détecter tous les paquets de demande de connexion vers la cible. Dans notre cas, les critères de ces paquets sont:

- paquets de type TCP
- provenant de n'importe quelle source
- à destination de la cible 192.168.1.17 et du port 80
- paquets vérifiés quelque soit l'état de la connexion (flow: stateless)
- paquets dont le flag SYN est positionné (flags: S)

Grâce à cette règle, Snort supprime tous les paquets correspondant aux critères précités. Cependant, la plupart de ces paquets sont probablement licites, car appartenant au trafic normal. Etant donné que le but est de détecter un comportement anormal, c'est-à-dire une grande affluence de demande de connexions, nous avons ajouté à notre règle la définition d'un seuil de tolérance :

drop tcp any any -> any 80 (msg : "Attention attaque synflloding"; flow: stateless; flags: S; detection_filter: track by_dst,count 10,seconds 1; sid:1000001;)

Nous avons donc défini ce seuil grâce au paramètre « detection_filter ». Celui-ci possède les options suivantes :

- count: spécifie le nombre d'occurrences provoquant l'alerte
- seconds: spécifie l'intervalle de temps
- track: « by_dst », vérifie le nombre d'occurrences par destination

Le choix des paramètres du seuil

Le choix des paramètres se révèle fort important, d'autant plus qu'ils sont étroitement liés. En effet, il existe 2 politiques :

- la première consiste à minimiser la valeur des paramètres, c'est-à-dire à choisir un nombre d'occurrences très petit pour un intervalle de temps peu élevé. Dans ce cas, l'attaque est très

rapidement détectée. Malheureusement, l'emploi de cette politique soumet la cible à une attaque consistant à envoyer un nombre de paquet légèrement inférieur au seuil pendant l'intervalle de temps afin de ne pas déclencher l'alerte. Le timeout de connexion étant très certainement supérieur à l'intervalle de temps, ces connexions resteront à demi ouvertes et continueront à consommer de l'espace mémoire. La difficulté de cette attaque pour le piratage est d'apprécier la valeur exacte de ces paramètres.

- La seconde politique consiste à choisir un intervalle de temps supérieur ou égal au timeout des connexions. Une attaque est alors détectée assez tard car le nombre d'occurrences doit être suffisamment élevé pour correspondre à ce grand intervalle de temps. En revanche, l'attaque précédente (première politique) ne pourra jamais se produire. Un autre avantage est qu'un pic momentané de connexions ne sera pas détecté comme une attaque.

Il faut trouver un juste équilibre entre ces deux politiques pour éviter les inconvénients qui leur sont liés, d'où toute la complexité du choix du seuil. Chaque administrateur devra se baser sur les caractéristiques de son réseau et sur les besoins de ses clients, ceux-ci fluctuant très souvent. Il lui faudra se mettre à jour régulièrement, cette technique de détection n'étant pas fixée.

La détection du seuil critique par le pirate

Une fois la règle du seuil correctement définie, il est logiquement impossible pour l'attaquant de créer un Syn Flood provoquant la non disponibilité du serveur sans être détecté. Le pirate devra donc choisir une alternative à son attaque : soit utiliser un autre type d'attaque, soit la faire évoluer. Quelle que soit l'évolution de l'attaque, il est fort probable que le pirate devra chercher à connaître le seuil de détection afin de ne pas se faire détecter.

Détecter ce seuil avec des moyens informatiques sans être détecté est pratiquement impossible car le pirate va devoir déclencher au moins une fois l'alerte pour observer les mesures en action lors de la détection. Par exemple, la mesure prise lors de la détection d'un flood de Syn est l'activation des Syn Cookies, le pirate va alors monter graduellement son attaque et chercher un signe de cette activation dans les réponses. Le plus simple est certainement de faire passer cette détection pour une fausse alerte ou un flood licite. Il reste bien sur la possibilité de profiter d'une faiblesse humaine pour obtenir ce renseignement.

Le pirate pourrait également vouloir connaître le seuil critique. Il est tout à fait possible de se servir de l'attaque afin de trouver la valeur exacte de ce seuil.

Pour connaître ces valeurs, nous avons utilisé notre attaque en envoyant un nombre arbitraire de paquets SYN au serveur web ; tout juste après, nous avons essayé de nous y connecter avec un navigateur afin de vérifier si de nouvelles demandes de connexions étaient toujours acceptées. Après une recherche graduelle, nous sommes arrivés aux valeurs citées précédemment.

Ensuite, nous avons voulu évaluer la valeur du timeout. Pour ce faire, nous avons envoyé un nombre de paquets SYN supérieur au seuil critique afin qu'aucune autre demande de connexion ne soit acceptée. Nous avons réalisé des demandes de connexion à l'aide du navigateur jusqu'à ce qu'elles soient acceptées, et nous avons déterminé un timeout d'environ 20 secondes, tant sous Windows que sur Linux.

Il est donc facile de déterminer les valeurs de ces variables au moyen de la seule attaque.

2.5.2. La technique de détection des connexions semi-ouvertes

L'inconvénient de la technique précédente est qu'elle ne différencie pas les demandes de connexions normales de celles provenant d'un pirate. La technique comptabilise le nombre de paquets Syn reçus, sans se soucier que la connexion ait été établie ou non. Une alerte a donc une

certaine probabilité d'être un « faux positif ».

Une étape supérieure serait de pouvoir détecter les demandes de connexions résultant d'une attaque. Ce qui caractérise ces demandes est que les connexions sont à demi-ouvertes et le restent. Les demandes de connexions normales sont généralement rapides et ne restent à l'état semi-ouvertes que très peu de temps.

Notre idée est donc de tenir en mémoire un compteur de ces connexions semi-ouvertes. Un seuil définit le nombre de connexions semi-ouvertes que le système tolère. Lorsque le compteur atteint le seuil, il est raisonnable de penser que l'on est sujet à une attaque et qu'une majorité des connexions semi-ouvertes en attente sont utilisées pour l'attaque.

Cette technique permettrait d'identifier plus précisément les connexions frauduleuses et de limiter fortement le nombre de fausses alertes. Il faut maintenant voir si le coût de mise en place est rentable : dans le cas d'utilisation d'un logiciel de détection tiers, cela peut s'avérer aussi coûteux de détecter l'attaque que de la subir car il faut maintenir à jour une table des connexions, ce qui est exactement ce que le système attaqué fait.

2.5.3. Des règles supplémentaires

La technique de détection des connexions semi-ouvertes peut s'avérer difficilement applicable. Il faut donc chercher une nouvelle orientation. Il s'agit de trouver ce qui caractérise un Syn Flood « frauduleux » (<> Syn Flood créé par une grande affluence de clients) ; nous ne cherchons maintenant plus à détecter les effets immédiats mais plutôt les effets secondaires.

Dans certains cas de figure, le pirate ne peut empêcher complètement la machine usurpée de répondre. Il reste donc certains paquets RST trahissant la présence d'une attaque. Notre règle détecte donc un seuil anormal de paquet RST :

```
alert tcp any any -> 192.168.1.14 80 (msg:"RST"; flow:stateless; flags:R; threshold: type both, track by_dst, count 100, seconds 20; sid:1000003;)
```

De même, si le pirate emploie une adresse qui ne soit pas présente sur le réseau, certains routeurs sont configurés pour répondre par un paquet ICMP de type « host unreachable » (il se peut que le routeur réponde aussi avec des RST, ce qui empêche la bonne réalisation de l'attaque, néanmoins il peut être intéressant de détecter cette tentative) :

```
alert icmp any any -> any any (msg:"Host unreachable"; itype: 3; icode: 1; threshold: type both, track by_dst, count 100, seconds 20; sid:1000002;)
```

Ces règles à elles seules ne suffisent pas mais elles permettent d'affiner la différenciation des bonnes alertes des mauvaises.

2.6. Réaction de l'attaquant

Une fois ces trois règles définies, l'attaquant n'a pas énormément de possibilités. S'il arrive à détecter le seuil, il pourrait rester en dessous afin de ne pas être détecté et simplement ralentir le trafic. Ceci ne sera plus valable dès qu'un trop grand nombre de clients se connectera, ce qui fera atteindre le seuil et donc déclencher l'alerte. De plus, ralentir le trafic n'est pas vraiment satisfaisant ; cela ressemblerait seulement à une période de plus forte affluence, chose à laquelle un serveur est préparé.

Connaître le seuil peut cependant lui permettre de générer des alertes semblant fausses afin de créer un relâchement au niveau de la surveillance.

3. Les honeypots

Dans le cadre de notre projet TER du second semestre, nous avons choisi de nous intéresser aux honeypots ou « pots de miel ».
Un honeypot est une ressource de l'architecture de sécurité dont le but est de se faire passer pour une cible réelle afin d'être attaquée ou compromise. Autrement dit les honeypots sont des machines de production destinées à attirer les pirates. Ceux-ci, persuadés d'avoir pénétrer le réseau, ont tous leurs faits et gestes contrôlés.
Les différentes implémentations des honeypots reposent sur leur niveau d'interaction. Le terme interaction désigne l'interaction entre le pirate et le système piraté. Les honeypots sont principalement divisés en deux catégories : les honeypots à faible interaction et les honeypots à forte interaction :

- Les honeypots à faible interaction sont les honeypots les plus simples. Ils ne fournissent pas de véritables services, ils se contentent de les simuler par l'intermédiaire de script comme Honeyd le propose et que nous allons utiliser par la suite.
- Les honeypots à forte interaction par contre fournissent de vrais services sur une machine plus ou moins sécurisée. Néanmoins les risques sont très nombreux puisque la machine est très vulnérable. Il faut donc s'assurer que l'architecture sous jacente soit bien sécurisée.

3.1 Types de honeypots

Selon ce qu'on attend de lui, un honeypot peut être de production ou de recherche :

1) Honeypot de production
Ce type de honeypots a une utilité pour la sécurité active du système pour lequel il est installé. Il déroute les attaques orientées vers les différents services de production du système, en les attirant vers lui. Ainsi les honeypots de production réduisent le risque, en renforçant la sécurité qui est assurée par les autres mécanismes de sécurité comme les firewalls, les IDS (systèmes de détections d'intrusions).

2) Honeypot de recherche
Ce sont des honeypots dont le souci n'est pas de sécuriser un système particulier. Ils sont introduits dans un environnement de recherche pour comprendre et étudier les attaquants et leur façon de procéder. Les renseignements tirés vont servir pour améliorer les techniques de protection contre ces attaques.
Les deux types de honeypots jouent un rôle dans une ou plusieurs composantes de la sécurité qui sont la prévention, la détection, et le recouvrement.
Les honeypots de production contribuent à la prévention du système, en provoquant une déception chez les attaquants, après plusieurs tentatives échouées pour atteindre les ressources du système. Et ils sont aussi bénéfiques pour la détection, dans la mesure que toute connexion établie avec un honeypot de production est considérée comme tentative d'intrusion au système, il élimine ainsi toutes les fausses alertes (positives et négatives). Le rôle des honeypots de production dans le recouvrement se traduit par les deux points suivants :

- premièrement, ils permettent une continuité des services après une attaque produite en leur sein, en les mettant simplement hors service.
- deuxièmement, l'information enregistrée par les honeypots de production sera d'un apport considérable pour le recouvrement du système.

Les honeypots de recherche ne servent pas la sécurité des systèmes (prévention, détection et

recouvrement) d'une manière directe, mais ils offrent des renseignements précieux sur les attaquants et leur comportement. Ces informations permettent une meilleure connaissance de la communauté des attaquants (blackhat community), ce qui aide les professionnels de la sécurité informatique dans l'amélioration de méthodes et mécanismes de protection.

3.2 Les honeynets

Un honeynet est un réseau de systèmes honeypots et un ensemble de mécanismes de sécurité comme les firewalls, les IDS, les serveurs log, etc. Il donne apparence à tout un environnement de production avec des failles et des informations pertinentes utilisées comme appât pour attirer et piéger les attaquants. Toutes les actions de la victime seront surveillées et enregistrées. Sa structure de réseau permet même d'avoir des renseignements sur les communications entre les attaquants et leurs méthodes de collaboration.
Le fonctionnement d'un honeynet se décompose en trois opérations : le contrôle de données, la collecte de données et l'analyse de données.

- Le contrôle de données signifie un contrôle de toutes les données dans le système, mais surtout le trafic sortant (outbound traffic) qui peut contenir des attaques envers d'autres systèmes réels. Cela est assuré par exemple par un firewall dissimulé derrière un routeur, qui utilise plusieurs techniques comme le blocage du trafic après un nombre limité de connections. Ainsi le routeur représente une deuxième couche de contrôle du trafic sortant.
- La collecte de données désigne la capture de toutes les informations qui se rapportent sur l'attaquant et ses activités. Elle est effectuée sur trois niveaux : le firewall, les IDS, et les systèmes honeypots eux même.

L'analyse de données est l'opération qui consiste à extraire les informations recherchées, telles que les outils, les méthodes et tactiques utilisées par les hackers, ou bien celles qui révèlent les vulnérabilités existantes dans le système. C'est à partir des données collectées pendant l'attaque, et après leur analyse qu'on parvienne à avoir ce type d'informations recherchées sur le pirate et le système piraté.
Afin qu'il soit efficace et rentable pour l'organisation l'utilisant, un honeypot doit être, constamment, surveillé, maintenu, et mis à jour. Car il est en exposition permanente aux attaques qui ne sont pas toutes faciles à contenir, surtout que les attaquants essaient toujours de surpasser les mesures de sécurité mises en place.

3.3 Honeynets virtuels

Un honeynet virtuel est un honeynet dont tous ses systèmes honeypots sont installés sur une même machine qui fonctionne sous un système d'exploitation hôte. Le mot virtuel vient du fait que chaque système donne apparence qu'il est installé dans sa propre machine indépendante.
Les honeynets virtuels peuvent être indépendants ou hybrides. Un honeynet virtuel indépendant regroupe tout ses composants sur un même système physique (typiquement un firewall à l'entrée et un ensemble de honeypots), ce qui réduit le coût et le délai de son développement et facilite sa connexion à l'importe quel réseau informatique. Tandis que un honeynet hybride est un hybride de honeynet classique et de honeynet indépendant : les éléments de capture de données (exemple : firewall) et les éléments de contrôle de données (exemple : IDS) sont installés sur des systèmes isolés, et les honeypots sont regroupés sous un même système. Cette dernière solution est plus sûre, du moment que l'attaquant ne peut pas s'emparer du firewall et de l'IDS, et elle présente l'avantage de supporter une multitude de types de logiciels et de matériel.
Entre autres plusieurs technologies qui permettent l'implémentation d'un honeynet virtuel, on cite : VMware et User Mode Linux (UML)

- VMware est un exemple d'infrastructure logicielle pour le déploiement d'un honeynet virtuel. Il permet à partir d'un système d'exploitation principal appelé système hôte (HostOS) d'installer d'autres systèmes d'exploitation émulant des machines virtuelles appelés systèmes clients (GuestOS). Il existe deux possibilités de connexion d'une machine VMware : la première appelé bridge networking dans laquelle le système hôte peut paraître comme l'une des machines virtuelles et chaque machine virtuelle a une liaison directe au réseau. La deuxième consiste à connecter que le système hôte au réseau (host only networking) et chaque machine cliente n'est accédée que via ce système hôte.
- User Mode Linux (UML) est un dispositif open source développé par Jeff Dike, permettant d'exécuter plusieurs versions virtuelles de Linux, en même temps, sur un même système. Ainsi, il permet d'avoir plusieurs machines virtuelles sur une seule machine physique hôte exécutant Linux. UML est capable de créer plusieurs réseaux virtuels et même des routeurs virtuels, qui seront tous dans le réseau virtuel original. Du même que VMware, UML supporte les deux possibilités de connexion : bridge networking et host only networking.

3.4 Avantages et inconvénients des honeypots

1) Avantages

Le but d'un honeypot est de capturer un pirate et/ou de l'occuper pendant un certain temps, temps pendant lequel il ne s'attaquera pas aux vrais systèmes de production. C'est dans cette optique qu'il est important de rappeler que les honeypots ne sont pas une solution que l'on place pour résoudre un problème mais un outil à exploiter. Ce sont des alliés très efficaces pour les IDS et sont des solutions très rapides à mettre en place.

Enfin, utiliser un honeypot au sein d'un réseau interne d'une entreprise se révèle être un outil redoutable pour la détection des actes de malveillance provenant de l'intérieur.

2) Inconvénients

Puisque les honeypots doivent simuler des services et des systèmes utilisés par les vrais systèmes de production, ils doivent être attractifs afin de susciter l'intérêt sinon il sera ignoré donc par conséquent inutile.

3.5 Quelques solutions existantes

Les solutions décrites ci-après représentent des exemples pris d'une très longue liste de honeypots disponibles actuellement.

- BackOfficer Friendly (BOF), un honeypot simple à faible interaction, développé par Marcus Ranum, fonctionne dans un environnement graphique (Windows ou Unix), il émule quelques services de base comme : http, ftp, Telnet, mail, ou Back Orifice. BOF est capable de détecter toute tentative de connexion ou de scan des ports, et il a l'option « faking replies » qui consiste à donner à l'attaquant des fausses réponses.
- Honeyd est une solution honeypot à faible interaction. C'est un système OpenSource développé par Niels Provos, il fonctionne sur les systèmes Unix, et porté même sur Windows. Ce honeypot simule des services et même des systèmes d'exploitation réels sur des adresses IP non utilisées d'un réseau. Très complet et simple d'utilisation, grâce à ses fichiers de configuration, Honeyd vous permet d'émuler un grand nombre de services, de personnaliser les réponses aux connections, de simuler différentes stack ip pour tromper l'attaquant sur la version du système d'exploitation. Pour utiliser Honeyd, les trois modules

libevent, libdnet et libpcap sont nécessaires.

- Mantrap est un produit très complet proposé par Symantec. Ce honeypot est de haute interaction qui met en œuvre quatre systèmes d'exploitation logiquement séparés et qui sont installés sur une même machine hôte. Chacun de ces systèmes supporte des applications réelles, et apparaît comme système indépendant avec sa propre interface réseau. L'administration du système hôte se fait par une interface utilisateur graphique Java. Mantrap peut être utilisé comme honeypot de production, notamment pour la détection et la réaction, et il est aussi rentable dans la recherche, mais avec un risque considérable.

Nous utiliserons Honeyd dans le cadre de notre projet.

3.6 Honeyd

1) Test avec une commande ping

Après l'installation d'honeyd (voir annexe), nous allons lancer Honeyd en mode interactif pour vérifier si un de nos hôtes configurés répond à la commande ping. Lançons Honeyd en mode interactif. Pour ce faire taper dans la console :

```
honeyd -d -p /etc/honeypot/nmap.prints -l /var/log/honeypot/honeyd.log -f /etc/honeypot/honeyd.conf -i lo 10.0.0.0/8
```

Maintenant dans un autre terminal nous allons tenter de pinguer un de nos hôtes configurés. Pour cela il suffit de taper la commande suivante :

```
ping 10.3.0.1
```

Nous nous rendons compte que Honeyd a bien reçu notre ping grâce au Reply qu'on voit au niveau du terminal où on a lancé notre Honeyd :

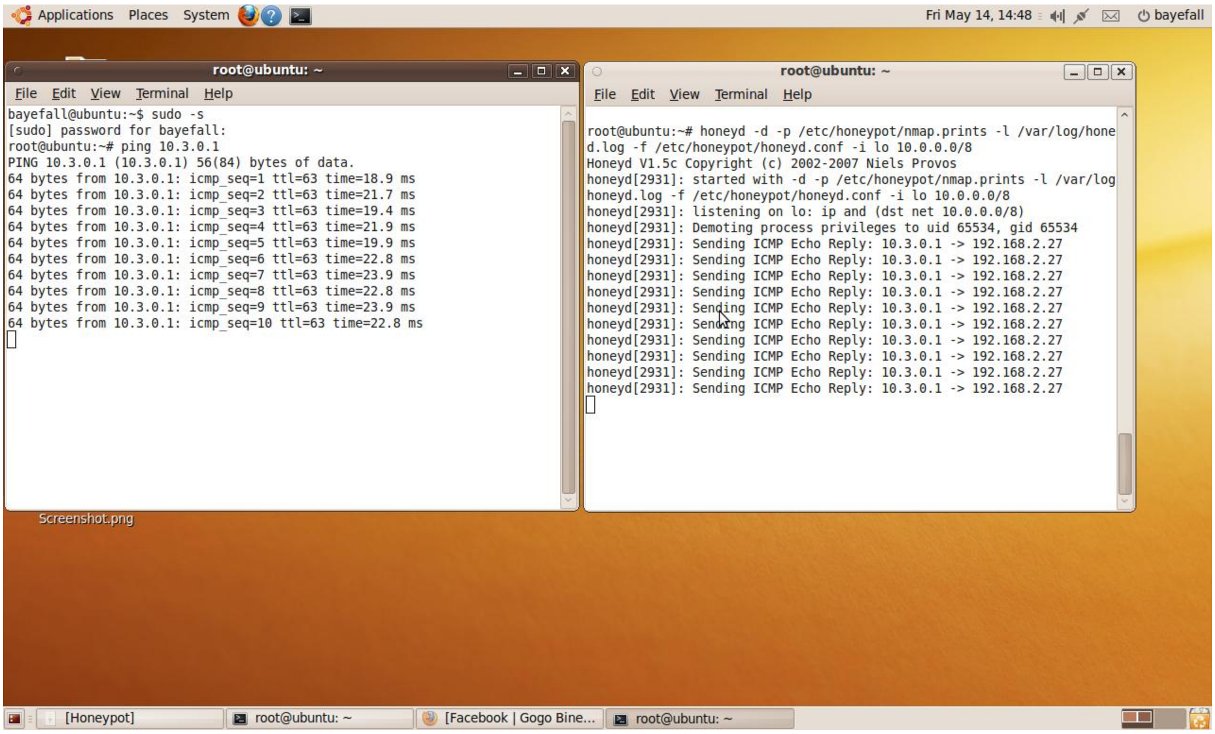


Figure 5 : capture d'écran pour le fct° de Honeyd.

2) Test avec une commande ping

Honeyd permet l'utilisation de scripts pour émuler un service fonctionnant sur une machine virtuelle. Ce service peut être un service telnet, ftp etc. Ces scripts peuvent être écrit en langage PERL ou même directement en SHELL. Des exemples de scripts sont fournis avec l'installation de Honeyd. Ils sont disponibles dans le répertoire scripts disponibles à l'adresse suivante:

/usr/share/honeyd/scripts

Il est possible de trouver de nouveaux scripts sur le site officiel à l'adresse suivante : http://www.honeyd.org/contrib.php Il est important de ne pas oublier tous les droits d'exécutions aux nouveaux scripts après leurs téléchargements.

A présent nous allons lancer Honeyd et vérifier si un de nos hôtes que nous avons configurés répond à l'appel d'un script. Nous avons choisi d'émuler un service Telnet donc nous tenterons d'accéder à un de nos hôtes configurés par le port 23.

honeyd -p /etc/honeypot/nmap.prints -l /var/log/honeypot/honeyd.log -f /etc/honeypot/honeyd.conf -i lo 10.0.0.0/8

Dans une autre console nous allons tenter d'accèder à un de nos hôtes sur le port 23. Pour cela taper dans la console :

telnet 10.3.0.1 23

Nous pouvons nous rendre compte qu'effectivement Honeyd a bien reçu notre tentative d'accès au port 23. Honeyd est entièrement mis en place il est alors possible de leurrer des attaquants potentiels.

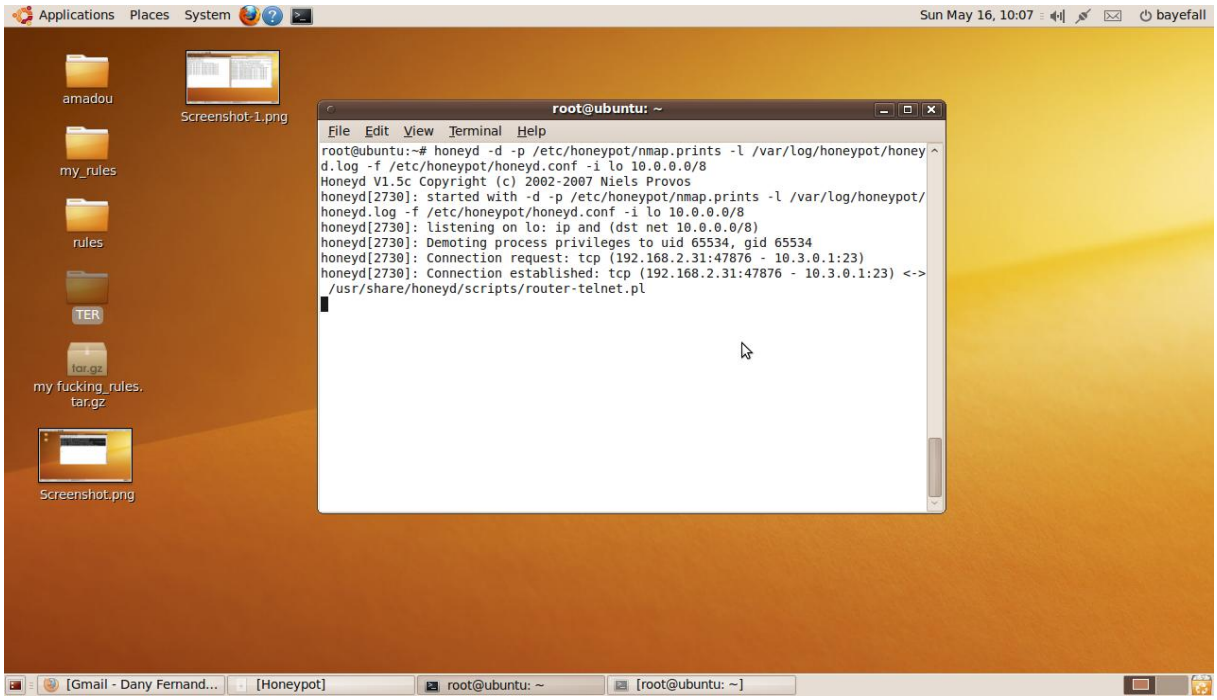


Figure 6 : capture d'écran pour le Honeyd avec une tentative de telnet.

3) Résultat

Grâce à son système de script nous pouvons aisément nous rendre compte que Honeyd est un honeypot très souple. Attention tout de même car Honeyd n'a pas été conçu pour fonctionner dans un système de production mais plutôt dans le domaine de la recherche pour pouvoir améliorer à l'avenir la sécurité d'un réseau.

4. Conclusion

Les attaques par déni de services existent depuis de nombreuses années et sont parfois médiatisés. Elles ont su se développés au fur et à mesure du développement des systèmes informatiques, tout en étant simple à mettre en œuvre. Bien qu'aujourd'hui, il existe des outils de détections et de protection contre le déni de service, ils sont toujours complexes à mettre en place avec une efficacité pas toujours optimale. Avoir une multitude d'outil ne suffit pas, il faut surtout être réactif lorsque l'on subit une attaque, mais aussi avoir une bonne politique de sécurité, non pas pour supprimer totalement les risques, car le risque zéro n'existe pas, mais au moins limiter l'impact d'une attaque de type déni de service face au service que l'on protège. Mais aujourd'hui, les pirates ont accès à des ressources de plus en plus importantes, leurs attaques sont donc de plus en plus difficiles à contrer. La lutte contre les pirates n'est donc pas prête de s'arrêter.

Références

http://www.honeynet.org/

http://www.ubuntu-fr.org/

http://openmaniak.com/

http://www.ouah.org/hpin2.htm

http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html

http://www.symantec.com/connect/fr/articles/hardening-tcpip-stack-syn-attacks

http://www.linux-france.org

http://www.commentcamarche.net

http://tomicki.net/syn.flooding.php

http://emergingthreats.net/

http://www.student.montefiore.ulg.ac.be/

Annexes

Installation de snort :

1ère étape – Installation de Libnet
Libnet est une librarie requise par snort. La version de snort utilize requiert la version 1.0.x de Libnet et nous avons choisi la version 1.0.2a :

```
cd /usr/src
wget http://www.packetfactory.net/libnet/dist/deprecated/libnet-1.0.2a.tar.gz
tar xzvf libnet-1.0.2a.tar.gz
cd Libnet-1.0.2a
./configure
make
make install
```

2ème étape : installation de snort
L'option enable—inline de la configuration est requise pour activer le mode inline de snort et l'utiliser comme un IPS :

```
cd /usr/src
wget http://www.snort.org/dl/current/snort-2.4.5.tar.gz
tar xzvf snort-2.4.5.tar.gz
cd snort-2.4.5
./configure --enable-inline --with-libipq-includes=/usr/include/libipq
make
make install
```

Etape 3 : configuration de l'installation de snort
Utilisez les commandes suivantes pour créer les dossiers contenant les fichiers de configuration, les règles, et les fichiers de log utilisés par snort.

```
mkdir /var/log/snort
mkdir /etc/snort
mkdir /etc/snort/rules
cp /usr/src/snort-2.4.5/etc/* /etc/snort
```

Editez le fichier /etc/snort.conf avec la commande

```
vi /etc/snort/snort.conf
```

Modifier le fichier snort.conf pour que la variable RULE_PATH référence le nouveau fichier de stockage des règles

```
# Path to your rules files (this can be a relative path)
# Note for Windows users: You are advised to make this an absolute path,
# such as: c:\snort\rules
var RULE_PATH /etc/snort/rules
```

L'installation de snort est maintenant terminée mais à ce stade, snort n'a aucune règles avec lesquelles fonctionner. Le téléchargement des règles nécessite l'installation de l'outil Oinkmaster.

Installation d'Oinkmaster

Il est nécessaire d'installer les règles de signature Snort et de les maintenir à jour. Il existe plusieurs site maintenant des liste de signature.

```
apt-get install oinkmaster
```

Site officiel de snort (http://www.snort.org) : Pour télécharger les règles Snort, nous avons besoin de créer un compte gratuit sur le site web de Snort. Une fois connecté avec votre compte Snort, vous pouvez obtenir un code en bas de page.

Ouvrir le fichier /etc/oinkmaster.conf :

```
sudo gedit /etc/oinkmaster.conf
```

Modifiez le paramètre "url" :

```
url = http://www.snort.org/pubbin/oinkmaster.cgi/code/snortrules-snapshot-2.8.6.tar.gz
```

Ajoutez la ligne suivante:

```
modifysid * "^alert" | "drop"
```

Création du dossier de sauvegarde.

```
mkdir /etc/snort_inline/backup
```

Ajouter un utilisateur appelé oinkmaster :

```
useradd oinkmaster
```

Définir les permissions :

```
chown -R oinkmaster /etc/snort_inline/backup
chown -R oinkmaster /etc/snort_inline/rules
chown -R oinkmaster /var/run/oinkmaster
chmod 644 /etc/snort_inline/snort_inline.conf
```

Teste du script oinkmaster :

```
su oinkmaster
oinkmaster#oinkmaster -o /etc/snort_inline/rules -b /etc/snort_inline/backup 2>&1
```

Mise à jour automatique (chaque jour à 00:30) :

```
crontab -e -u oinkmaster
30 00 * * * oinkmaster -o /etc/snort_inline/rules -b /etc/snort_inline/backup 2>&1
```

Installation de Honeyd

Pour installer Honeyd, nous avons utilisé Aptitude, en tapant la commande suivante : apt-get install honeyd. Voici les principaux fichiers installés :

/etc/init.d/honeyd
/etc/logrotate.d/honeyd
/etc/default/honeyd
/usr/lib/honeyd
/usr/share/honeyd
/usr/share/doc/honeyd
/usr/include/honeyd
/usr/bin/honeyd

Nous tenons à rappeler qu'on a utilisé Ubuntu 9.10 qui est une distribution GNU/Linux libre et gratuite.

L'installation de Honeyd va aussi installer les dépendances suivantes :

-libpcap0.8
-rrdtool
-librrd4
- libdumbnetl
-farpd
-honeyd-common

La libpcap est une bibliothèque de fonctions servant d'interface à la capture de paquets et est indépendante du système.

RRDtool est un outil de gestion de base de données à RRD permettant la sauvegarde haute performance et le tracé de graphiques, de données chronologiques.

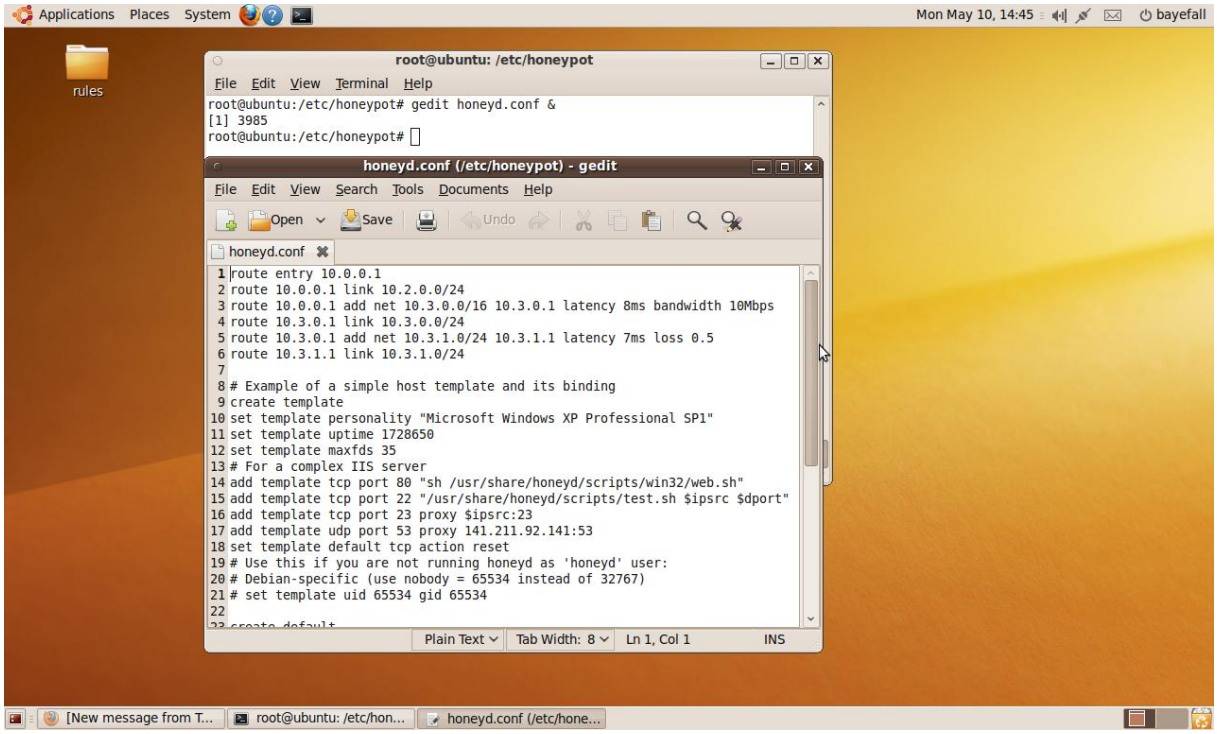
Libdumbnetl est une bibliothèque réseau, portative qui fournit une interface simplifiée pour plusieurs routines réseau

Farpd est un démon ARP répondant à n'importe quel démon ARP d'un ensemble d'adresses Ip.

1- Fichier de configuration

Le fichier de configuration utilisé est similaire au fichier installé par défaut. Pour le

visualiser c'est simple. IL suffit de préciser dans la console avec quel éditeur on souhaite le visualiser (gedit par exemple) et préciser le chemin du fichier. Voici la commande : gedit /etc/honeypot/honeyd.conf



2- *Mise en place*

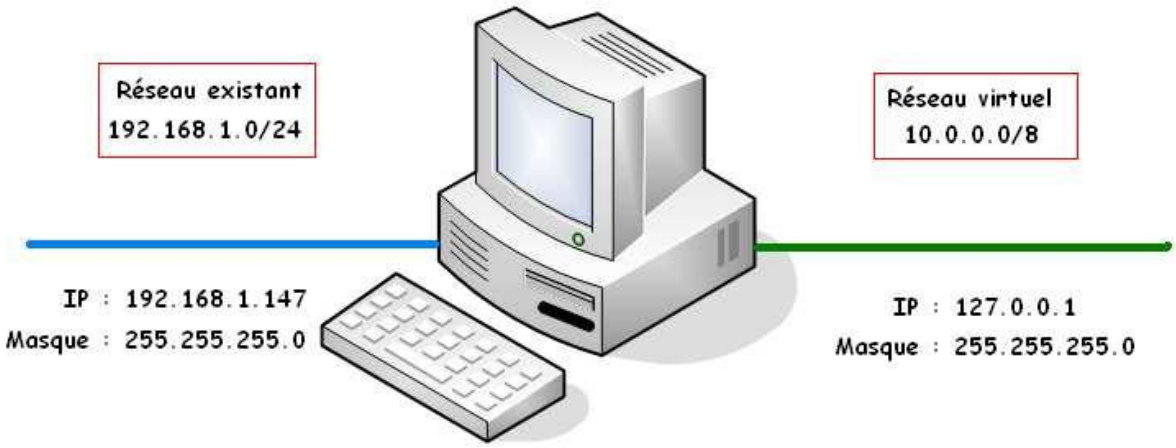
Dans cette section nous allons utiliser la configuration par défaut proposée par Honeyd dans le fichier honeyd.conf visualiser précedemment.

```
route entry 10.0.0.1
route 10.0.0.1 link 10.2.0.0/24
route 10.0.0.1 add net 10.3.0.0/16 10.3.0.1 latency 8ms bandwidth 10Mbps
route 10.3.0.1 link 10.3.0.0/24
route 10.3.0.1 add net 10.3.1.0/24 10.3.1.1 latency 7ms loss 0.5
route 10.3.1.1 link 10.3.1.0/24
```

Pour faire fonctionner ce réseau virtuel ci-dessus il faudra au préalable déclarer une route réelle dans la table de routage pour l'atteindre. La passerelle utilisée pour cette route sera l'interface lookpack (localhost) afin de ne pas perturber le réseau existant. Pour cela il faudra taper dans la console la commande suivante :

```
route add -net 10.0.0.0 netmask 255.0.0.0 gw localhost
```

Voici un shema montrant la configuration actuelle :



Exécution

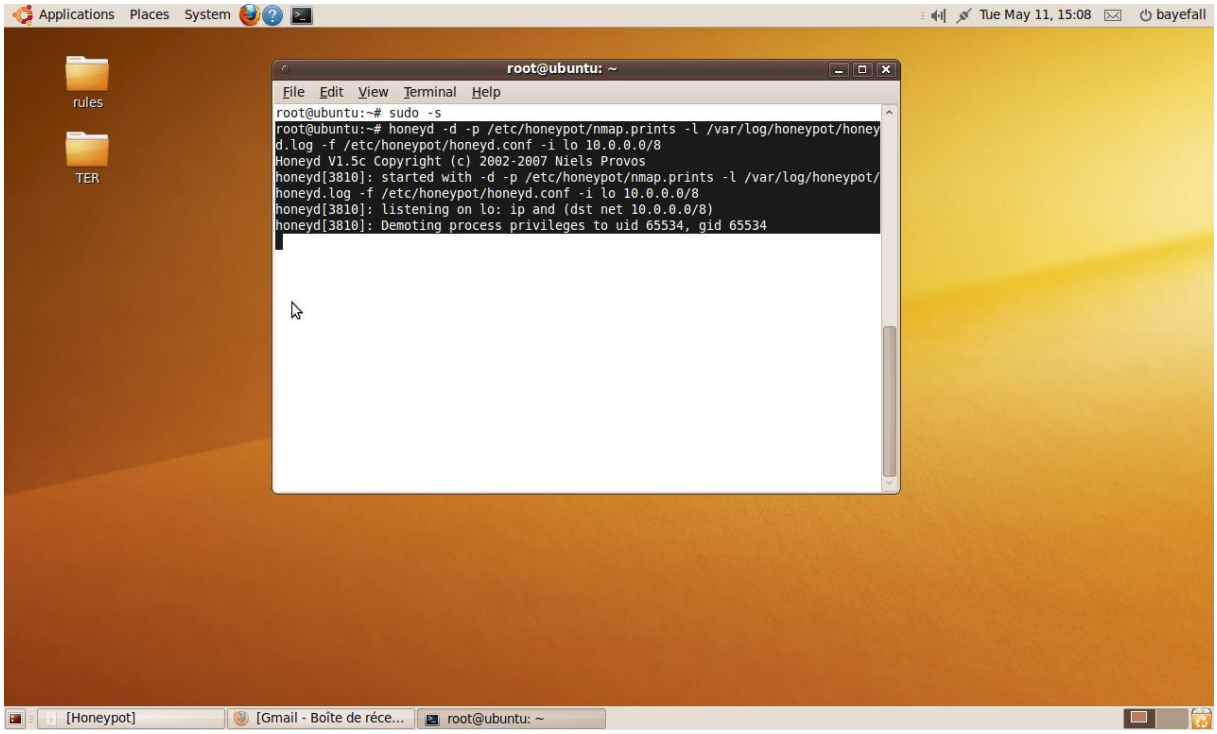
L'exécution de fait à l'aide la commande qui suit dans notre console :

```
honeyd -d -p /etc/honeypot/nmap.prints -l /var/log/honeypot/honeyd.log -f
/etc/honeypot/honeyd.conf -i lo 10.0.0.0/8
```

Attention : Avant l'exécution il est nécessaire de créer le fichier honeyd.log et de lui administrer tous les droits. Ceci est possible en tapant dans notre console la commande suivante :

```
touch honeyd.log
chmod 777 honeyd.log
```

Les détails des paramètres de l'exécution :
-d signifie qu'on lance notre honeyd en mode interactif
-p fichier des fingerprints
-f fichier de configuration



Voici ce qu'on voit à l'écran. Ceci signifie que tout fonctionne bien. Faisons un ctrl+C pour arrêter la commande. Puis nous allons maintenant lancer notre démon honeyd. Mais avant cela nous

allons éditer et modifier le fichier de configuration du démon.

Pour ce faire nous allons éditer le fichier à l'aide de la commande suivante :

```
gedit /etc/default/honeyd
```

Dans ce fichier de configuration, il faudra modifier la commande RUN en l'attribuant la valeur «yes » afin de pouvoir démarrer le démon :

```
RUN= ‘‘yes’’
```

Il faut aussi indiquer l'interface réseau à utiliser et préciser la plage d'adresse réseau à utiliser :

```
INTERFACE="eth0"
NETWORK=10.0.0.0/8
```

Nous allons pouvoir enfin lancer notre démon Honeyd à l'aide de la commande suivante :

```
/etc/init.d/honeyd start
```

S'il ya pas d'erreurs on doit obtenir le message suivant :

```
* Starting Honeyd daemon honeyd
```