LANL - TP07

1 Introduction

Ce TP est consacré aux Firewall Linux.

Lorsqu'un paquet arrive sur un ordinateur par l'une de ses interfaces réseau, le premier programme qui reçoit les données est le driver de la carte réseau. Ce driver est en fait un bout de code du noyau (kernel) qui fait partie intégrante de ce dernier. en Linux, on appelle cela un *module kernel*. Le kernel et tous ses modules tournent avec des privilèges très élevés sur le proccesseur.

Tous les autres logiciels qui tournent sur l'ordinateur n'ont pas de tels privilèges. Si un logiciel veut utiliser un périphérique (écrire un fichier sur le disque dur, sortir un paquet sur le réseau), ils doivent en faire la demande au kernel via un appel système *(system call)* qui déclenche une *interruption*. En résumé, le kernel reste (en théorie) le seul maitre à bord et c'est lui qui se charge, entre autres, de toutes les communications avec les périphériques.

Tout ceci pour dire que lorsque un paquet arrive via une interface réseau, c'est le kernel qui en premier reçoit les données. Lorsqu'un paquet n'est pas autorisé à entrer sur le système, c'est le kernel lui même qui le bloque car il est en première ligne. Lors d'une attaque, c'est d'abord le kernel qui reçoit les données. Si le paquet est autorisé, alors il est dépaqueté correctement et transmis à l'application destinataire. Cette dernière est déterminée grace au numéro de port. Le firewall est donc implémenté au niveau du kernel.

Sous Linux, le module kernel qui gère le firewall s'appelle *Netfilter* et il est pilotés grace aux outils *iptalbes* (pour IPv4) et *ip6tapbles* (pour IPv6)

2 Présentation

L'architecture *Netfilter* organise le filtrage suivant une structure hiérarchique. Le plus haut niveau est la **table**, qui se compose de **chaines**, enfin chaque chaine se compose d'une liste de **règles**.

Les tables qui existent par défaut sont les suivantes :

— **filter** talble par défault, contient les chaines suivantes :

  — *INPUT* : concerne tous les paquets qui sont à destination de l'ordinateur

    - *OUTPUT* : concerne tous les paquets qui sont produits par l'ordinateur lui même
    - *FORWARD* : concerne tous les paquets qui sont en transit
- **nat** : table pour les règles de NAT, contient les chaines suivantes :
    - *PREROUTING* : pour modifier les paquets avant de prendre la décision de routage les concernant, utile par exemple lorsqu'on modifie l'adresse de destination *DNAT (destination NAT)*
    - *POSTROUTING* : pour modifier les paquets après avoir pris la décision de routage les concernant, utile par exemple lorsqu'on modifie l'adresse source *SNAT (source NAT)*
    - *OUTPUT* : règle de NAT pour les paquets qui sont produits par la machine elle même, les deux autres chaines concernent les paquets en transit
- **mangle** : cette table est utilisée pour faire des modifications plus avancées sur les paquets : modification de la *QoS*, augmentation artificelle du *TTL*, etc
- **raw** : Ici on à quasiment la possibilité de modifier les paquets en mode binaire

3 Concepts importants

3.1 Connection Tracking

Imaginez que vous souhaitez autoriser les connexions sortantes vers toutes destinations uniquement pour les protocoles *http (port 80/TCP)* et *https (port 443/TCP)*. Vous écrirez donc une règle qui aura la forme suivante :

- : chaine : *OUTPUT* ou *FORWARD*
- : destination : *0.0.0.0/0* (toutes les destinations)
- : protocole : *TCP*
- : destination port : *80* et *443*

Si un paquet remplit ces conditions, il sera autorisé à sortir. Le serveur de destination le reçoit et envoie une réponse. La réponse arrive chez nous en *INPUT*. Sera-t-elle autorisée ? Sinon, quelle règle faudrait il écrire ?

Pour que la réponse puisse entrer, la théorie impose qu'on écrive la règle suivante :

- : chaine : *INPUT* ou *FORWARD*
- : source : *0.0.0.0/0* (toutes les sources)
- : protocole : *TCP*
- : source port : *80* et *443*

Une règle d'entrée qui autorise n'importe qui (0.0.0.0/0) à entrer, à condition que le port source des paquets émis soit *80/TCP* ou *443/TCP* !!! Ce qui est un gros trou de sécurité.

C'est là que le *Connection Tracking* intervient. Son rôle consiste à laisser passer des paquets entrants si et seulement si ils font partie d'une **connexion**

préalablement autorisée et établie, généralement depuis l'intérieur vers l'extérieur.

Par exemple, un paquet provenant de *164.15.251.133*, source port *443/TCP* ne sera autorisé a entrer que si il est la réponse à un paquet émis vers *164.15.251.133*, port de destination *443/TCP*.

Le firewall doit donc garder en mémoire des informations sur ce qu'il a laissé sortir pour savoir si ce qui rentre correspond. Cela s'appelle le *Connection Tracking*. On dit alors que le firewall est **Statefull**, il a connaissance des connexions en cours. par oposition à *Stateless*.

Remarque : on peut utiliser un routeur pour faire du filtrage via des *ACL*. la différence entre les *ACL* de routeur et un vrai firewall est que les *ACL* sont *Stateless*.

3.2 Evaluations des règles

Les règles sont évaluées dans l'ordre. de la première à la dernière. Dès que le paquet en cours d'évaluation correspond à une des règle, il est traité *(Jump)*. Si la première règle de la liste dit : bloquer tout, plus rien ne passe, même si il y à des ouvertures ensuite.

3.3 Default Policy

Pour chaque chaine, on peut définir une police par défaut qui concerne tous les paquets qui ne corresponde à aucune règle *(no matching rules)*. Cette police peut être réglée par exemple à *DROP*.

3.4 DROP ou REJECT

Refuser un paquet peut se faire de deux manières :

— *REJECT* : on détruit le paquet et on informe la source via un paquet *ICMP* que la destination n'est pas joignable, c'est poli mais ça dévoile le fait qu'il y à une machine qui fait du filtrage sur le chemin.

— *DROP* : on détruit le paquet et on informe personne. Ainsi l'expéditeur ne peut pas déterminer la cause : est ce qu'il y à un firewall ? est ce que la machine de destination est éteinte ? existe t'elle vraiment ? ...

4 Commandes utiles

— : afficher toutes les règles (rappel : si on ne précise pas la table (-t) c'est la table filter) :

```
iptables -nvL
```

```
iptables -t nat -nvL
```

— activer le *connection tracking* :

```
iptables -I INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Ici on définit un *match (-m) qui accepte tous les paquets liés à une connexion sortante* RELATED *ou qui s'inscrivent dans une connexion sortante préalablement établie* ESTABLISHED

— modifier la police par défaut d'une chaine (ne fonctionne que pour les chaines par default de la table) :

```
iptables -P INPUT DROP
```

— : ajouter une règle à la fin de la liste *(APPEND)* :

```
iptables -A INPUT -p tcp -s 15.32.23.0/24 --dport 22 -j ACCEPT
```

Ici on autorise toutes les connexion entrantes depuis le réseau *15.32.23.0/24* à destination du port *22/TCP*

— supprimer une règle :

```
iptables -D INPUT -p tcp -s 10.210.2.2/23 --dport 22 -j ACCEPT
```

— insérer une règle à une position donnée :

```
iptables -I INPUT 10 -p tcp -s 10.210.2.1/24 --dport 22 -j ACCEPT
```

Ici on insère une règle à la 10ème position de la chaine *INPUT*. Si le numéro n'est pas précisé, la règle est ajoutée en début de chaine.

— créer une chaine :

```
iptables -N TRUST
```

Ici on crée une chaine qu'on nomme TRUST.

— envoyer des paquets dans une chaine :

```
iptables -I INPUT -s 10.210.0.0/16 -j TRUST
```

— : ajouter une règle dans une chaine :

```
iptables -I TRUST -p tcp -m tcp --dport 22 -j ACCEPT
```

— supprimer toutes les règles :

```
iptables -F

iptables -X TRUST

iptables -X

iptables -t nat -F
```

L'option -F supprime toutes les règles des chaines par défaut, l'option -X supprime toutes les chaines ajoutées. Pour supprimer une chaine, il faut d'abord supprimer toutes les règles qui envoient des paquets vers cette chaine.