ATELIER NETFILTER : LE FIREWALL LINUX EN ACTION

Olivier ALLARD-JACQUIN olivieraj@free.fr

Version 0.9.1 - 20 avril 2004

Ce document est le support écrit d'un atelier / TP (Travaux Pratiques) que j'ai présenté le 1er avril 2004 dans le cadre de l'association Linux Guilde. Il s'agit d'une succession d'exercices et de leurs réponses afin d'étudier le comportement et la mise en oeuvre de Netfilter, le firewall de Linux. Vous trouverez les notions théoriques de cet atelier dans cette autre documentation que j'ai écris.

Vous trouverez la dernière version de ce document :

- sur son site principal : http://olivieraj.free.fr/fr/linux/information/atelier_netfilter/.
- sur le miroir de la Guilde.
- en version PDF (sans les scripts des solutions). Pour la lire, utilisez le lecteur Xpdf ou Acrobat Reader.
- sous forme d'archive complète, au format .tgz (.tar.gz), afin de la lire en hors ligne.

Pour les personnes désirant reproduire cet atelier chez eux, vous aurez besoin :

- De télécharger le fichier de la présentation au format OpenOffice.org. Vous pourrez alors suivre la présentation tout en faisant les exercices, en vous y référant à chaque fois que vous verrez la balise :

  Illustration : xxxx

- D'une machine sur laquelle sera installée un serveur web (Apache par exemple).
  - Publiez dessus la présente présentation. Pour cela, vous pouvez télécharger l'archive au format .tgz.
  - Cette machine devra avoir pour adresse IP 192.168.1.50. Si vous voulez utiliser une autre adresse IP, il faudra que vous modifiez les lignes "IP_SERVEUR=" dans les scripts "filter1a.sh", "filter2a.sh", "log1a.sh", "log2a.sh". Modifiez aussi les lignes "IP_MACHINE_*=" dans le script "userchaine2a.sh".
  - En terme de sécurité, cette machine devra accepter toutes les connexions du réseau local (192.168.1.0/255.255.255.0 dans notre exemple).

  Dans la suite de la documentation, cette machine sera appelée "machine du présentateur" ou "serveur".

- D'une seconde machine, reliée en réseau avec la précédente, sur laquelle vous allez travailler.
  - Elle devra utiliser le même réseau IP que la machine serveur.
  - Cette machine devra bien entendu tourner avec un kernel 2.4.x ou supérieur, être compilé pour utiliser Netfilter (le firewall de Linux), et posséder le iptables associé.
  - Il faudra aussi un navigateur web (Mozilla, ...), votre éditeur de texte favori, nmap (scannaire de ports réseaux), et éventuellement le xterm " rxvt", si vous voulez utiliser "rxvt_log" afin d'analyser agréablement vos logs Netfilter.
  - Pour la partie IV il faudra installer dessus un serveur web (Apache, ...) et un serveur FTP (ProFTPD, ...).

- Enfin, pour certains tests et démonstrations de l'atelier, vous aurez besoin d'une troisième machine configurée comme la 2nd machine. Cependant, c'est optionnel. Aussi si vous n'avez pas de 3ème machine disponible, vous ne ferez pas certains exercices et passerez aux exercices suivants.

Plan :

Introduction

Dans cet atelier, nous allons revoir quelques rapides rappels sur Netfilter, et quelles bases des techniques de firewall sous Linux. Puis nous mettrons en oeuvre des techniques simples de filtrages. Dans une 3ème partie, nous aborderons des filtrages plus évolués à usage tant personnels que professionnels. Nous aborderons enfin la mise en oeuvre de protections destinés aux serveurs.

I Rappels sur Netfilter

Le contenu théorique de cet atelier se base sur une documentation que j'ai écris, et que vous pouvez retrouver sur http://olivieraj.free.fr/fr/linux/information/firewall/.

Netfilter est une couche logiciel intégrée dans le kernel de Linux, c'est à dire le coeur même du système d'exploitation Linux. Elle se situe plus particulièrement au niveau des couches réseaux IP (du protocole réseau IP, appelé aussi par abus de langage "TCP/IP"), et se caractérise sous la forme de 5 hooks ("crochets" en Français) placés à 5 endroits stratégiques du système réseau du kernel.

Lorsque qu'un paquet IP arrive à un de ces hook, il va subit un certain nombre de vérifications (règles) que constituent la chaîne associé au hook en question. En fonction du résultat de ces vérifications, une action (appelée "cible" par la suite) sera décidé pour lui : suppression du paquet, acceptation, modification, etc...

Illustration : Vue générale et hooks

I-1 Règles

Comme leur nom l'indiquent, les règles sont des critères sur le contenu des paquets.
Les critères peuvent être multiples :

- Interface source ou destination.
- Adresse IP source ou de destination.
- Port source ou de destination.
- Type de trame.
- Nombre de paquets.
- Paquet marqué par la table Mangle.
- Etc...

Suivant si le paquet répond ou non aux critères, le paquet sera dirigé vers une cible ou un autre.
Exemple de cible :

- DROP : Le paquet est détruit purement et simplement.
- REJECT : Le paquet est détruit, mais l'expéditeur en est informé.
- ACCEPT : Le paquet est autorisé à continuer dans le traitement de la pile IP. Mais une autre règle située après la règle qui a accepté ce paquet peut très bien finalement décider de le supprimer.
- LOG / ULOG : Le paquet est autorisé à continuer de passer, mais ses caractéristiques sont notées au passage.
- MASQUERADE : Le paquet va être modifié, afin de dissimuler (de masquer en fait) certaines informations concernant son origine.
- SNAT : L'adresse IP source est modifiée.
- DNAT : L'adresse IP de destination est modifiée.
- MARK : Le paquet est marqué en y attachant une information.
- Une chaîne utilisateur : Le paquet sera passé à une autre chaîne, définie par l'utilisateur. Nous verrons cela plus loin.
- Etc...

Illustration : Règles

I-2 Chaînes

Les chaînes sont de deux types : pré-définie ou utilisateur. Dans les deux cas, les chaînes sont des empilements de règles, que le paquet IP va parcourir. Une fois que toutes les règles ont été appliquées sur le paquet, et que celui-ci ne correspond à aucune règle, la cible par défaut de la chaîne sera utilisée. En général, il s'agira de DROP ou de ACCEPT.

Les chaînes pré-défines sont au nombre de 5, et sont toutes associées à un hook particulier :

Hook	Chaîne	Description
NF_IP_PRE_ROUTING	PREROUTING	A ce stade, le paquet est "brut de forme", c'est à dire qu'il n'a subit aucune modification par rapport à ce que l'interface réseau a reçu.
NF_IP_LOCAL_IN	INPUT	Ce "hook" est très intéressant, car à ce stade, le paquet est prêt à être envoyé aux couches applicatives, c'est à dire aux serveurs et aux clients qui tournent sur la machine. C'est un des points principaux sur lequel nous allons travailler.
NF_IP_FORWARD	FORWARD	"Forward" ("faire suivre" en français) est assez particulier. Ce "hook" voit passer des paquets IP qui vont transiter d'une interface réseau à une autre, sans passer par la couche applicative. Pourquoi diantre faire suivre un paquet entre 2 interfaces réseaux, comme par exemple entre "eth0" et "ppp0" ? En fait, c'est afin de permettre à Linux de se transformer en passerelle, une sorte de connexion entre diverses interfaces réseaux.
NF_IP_LOCAL_OUT	OUPUT	Ce "hook" est l'équivalent du "NF_IP_LOCAL_IN", sauf qu'il est exécuté après que les couches applicatives aient traités, ou générés, un paquet IP. Tout comme "NF_IP_LOCAL_IN", c'est un point que nous allons voir en détail.
NF_IP_POSTROUTING	POSTROUTING	C'est l'équivalent du "NF_IP_PRE_ROUTING" pour les paquets IP sortants de la couche IP. A ce stade, les paquets sont prêt à être envoyés sur l'interface réseau.

Les chaînes utilisateurs sont une particularité intéressante de Netfilter. Elles peuvent être appelées en temps que cible d'une règle (d'une chaîne dite "appelante"), et contenant elles-mêmes des règles. A la fin de la chaîne utilisateur, si aucun traitement n'a été effectué sur le paquet IP, celui-ci continuera son traitement dans la chaîne appelante.

Illustration : Chaînes

I-3 Tables

Comme son nom ne l'indique pas, Netfilter ne s'occupe pas qu'uniquement du filtrage des paquets IP. C'est une de ses tache, mais ce n'est pas la seule. Ainsi, Netfilter est conçue pour gérer 3 taches. Et à

chaque tache, une table a été définie. Les 3 taches sont :

- Filtrage, via la table FILTER
- Translation d'adresse, via la table NAT (Network Address Translation)
- Modification des paquets IP, via la table MANGLE

Une table est composée d'un ensemble de chaînes pré-définies, et d'un comportement particulier. Ainsi :

- La table FILTER est composée des chaînes :
  - INPUT
  - FORWARD
  - OUTPUT

  Son but est d'assurer le filtrage des paquets IP, c'est à dire d'autoriser ou non un paquet IP à accéder à une application, ou à en sortir. La table FILTER est aussi responsable de l'autorisation du transfert des paquets IP d'une interface réseau à l'autre, dans le cas du NAT.

- La table NAT est composée des chaînes :
  - PREROUTING
  - OUTPUT
  - POSTROUTING

  Elle est spécifiquement destinée à assurer le NAT. Alors que dans la table FILTER, la chaîne FORWARD permet de définir quelles sont les paquets autorisés à être NATé, la table NAT sera chargée de la modification des paquets, afin qu'ils soient correctement utilisable sur l'interface de destination.

- La table MANGLE est composée des chaînes :
  - PREROUTING
  - INPUT
  - FORWARD
  - OUTPUT
  - POSTROUTING

  Cette table est très particulière. Elle permet de modifier les paquets, en leur rajoutant ou supprimant certaines informations. Cette table est pour l'instant utilisé dans le cadre de la QoS (Quality Of Service), permettant de rendre le traitement de certains paquets plus prioritaires que d'autres.

Illustration : Tables

I-4 Iptables

Lorsque l'on parle de firewall sous Linux, iptables est LA commande à connaître. Ce programme est la 2nd partie principale de Netfilter. Contrairement au code de Netfilter inclus dans le noyau Linux, "iptables" est un programme "classique" (i.e : il fonctionne dans l'espace utilisateur) que l'on lance manuellement. Bien entendu, seul le root a les droits suffisant pour l'utiliser.

Iptables permet au root de configurer Netfilter, en lui indiquant quelles sont les règles à utiliser pour les différentes tables et chaînes. Il se lance plusieurs fois de suite, chaque lancement permettant de créer/supprimer/visualiser une règles de Netfilter.

Un descriptif complet des paramètres d'iptables serait inutile. La commande "man iptables" étant LA référence de toute documentation sur cette commande. Cependant, en voici un bref résumé des principaux paramètres :

iptables [table] [action] [critères] [cible]

Pour plus d'informations sur ces paramètres ou pour avoir plus d'explications sur les exemples, vous pouvez consulter ce chapitre de ma documentation de Netfilter.

- [table]
  Ce paramètre permet de définir sur quelle table la configuration va s'effectuer. Comme vu précédemment, il y a 3 tables : FILTER, NAT et MANGLE.

Paramètre	Description
-t [le nom de la table : filter, nat, mangle]	Spécifie la table sur lequel on va travailler.
	Exemple : iptables -t filter -X

Remarque :Ce paramètre "[table]" est optionnel. Si aucune table n'est définie (i.e : il n'y a pas de "[table]", "iptables" utilisera l'option par défaut "-t filter").

- [action]
  Ici, on va indiquer à "iptables" ce que l'on va faire :

Paramètre	Description
-Z	Réinitialise les compteurs des paquets d'octets passant par la table.
	Exemple : iptables -t mangle -Z
-F [PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING]	Suppression de toutes les règles pré-définies de la chaîne indiquée. Ou, si aucune chaîne pré-définie n'est indiquée, de toutes les règles pré-définies de la table.
	Exemple : iptables -t filter -F FORWARD
-X [Chaîne utilisateur]	Suppression de toutes les règles de la chaîne utilisateur indiqué. Ou, si aucune chaîne utilisateur n'est indiquée, de toutes les chaînes utilisateurs de la table.
	Exemple : iptables -t nat -X
-L [Nom de table : filter, nat, mangle]	Affiche la liste des chaînes et règles de la table spécifiée. Les paramètres "-n" et "-v" sont très utiles avec ce paramètre.
	Exemple : iptables -t nat -L -n -v
-N Nouvelle_chaîne_utilisateur	Crée une nouvelle chaîne utilisateur. Par la suite, des règles peuvent êtres ajoutées à ces chaînes, grâce au paramètre "-A"
	Exemple : iptables
-A Nom_chaîne	Rajoute une nouvelle règle à la fin de la chaîne "Nom_chaîne". "Nom_chaîne" peut être une chaîne pré-définie ou une chaîne utilisateur.
	Exemple : iptables -A INPUT -i lo -j ACCEPT
-D PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING Numéro_de_chaîne	Supprime la chaîne indiquée par "Numéro_de_chaîne".
	Exemple : iptables -D OUTPUT 5
-P Nom_chaîne Cible	Définie à "Cible" la cible par défaut de la chaîne "Nom_chaîne".
	Exemple : iptables -t filter -P INPUT DROP

- [critères]
  Cette partie permet de définir sur quels critères du paquet IP la règle va s'appuyer, afin de déterminer si oui ou non le paquet répond à la règle. On peut utiliser plusieurs critères en même temps, afin d'affiner au mieux l'application de la règle.

Paramètre	Description
-i Interface_réseau	Indique l'interface réseau par laquelle le paquet arrive.
	Exemple : iptables -A INPUT -i eth0 -j ACCEPT
-o Interface_réseau	Indique sur quelle interface réseau le paquet va sortir.
	Exemple : iptables -A OUTPUT -o eth0 -j DENY
-s Adresse_ou_plage_IP	Définie l' adresse ou la plage d'adresses IP dont sont issues le paquet.

	Exemple : iptables -A INPUT -s 192.168.0.100 -j ACCEPT
-d Adresse_ou_plage_IP	Définie l'adresse ou la plage d'adresses IP a qui sont destinée le paquet.
	Exemple : iptables -A FORWARD -d 10.0.0.0/255.0.0.0 -j REJECT
-p Protocole	Précise sur quel protocole la règles s'applique.
	Exemple : iptables -A INPUT -p icmp -j DROP
--sport Port(s)	S'utilise avec le paramètre "-p". Cette option définie le port source dont est issus le paquet IP. Le paramètre "Port(s)" peut être un chiffre ("80"), une suite de chiffres séparés par des virgules ("21,53,100") ou un nom de service ("ssh").
	Exemple : iptables -A INPUT -p tcp --sport 80 -j ACCEPT
--dport Port(s)	S'utilise avec le paramètre "-p". Cette option définie le port cible a qui est destiné le paquet IP. Le paramètre "Port(s)" peut être un chiffre ("80"), une suite de chiffres séparés par des virgules ("21,53,100") ou un nom de service ("ssh").
	Exemple : iptables -O OUTPUT -p tcp --dport 21,53,80 -j ACCEPT
-m Nom_module	Indique à Netfilter qu'il faut utiliser le module "Nom_module" afin d'analyser cette règle. Un module tout particulièrement intéressant est le module "state", qui permet le suivi de connexion (conntrack).
	Exemple : iptables -A OUTPUT -p tcp -m state --state ! INVALID -j ACCEPT
--state [!] NEW, ESTABLISHED, RELATED, INVALID	Dans le cas de l'utilisation du module "state", indique quel(s) statut la trame doit avoir afin d'appliquer la règle.
	Exemple : iptables -I INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

- [cible] Ce dernier paramètre est important, car il permet d'indiquer à Netfilter ce qu'il doit faire lorsque les critères de la règle s'appliquent au paquet IP.

Paramètre	Description
-j ACCEPT, DROP, REJECT, LOG, ULOG, etc...	Spécifie la cible lorsque la règle s'applique.
	Exemple : iptables -A OUTPUT -o eth0 -j ACCEPT

Illustration : Iptables

II Filtrages de base

Dans cette partie, nous verrons comment mettre en oeuvre des filtrages assez simples.

II-1 Principes généraux du filtrage

La technique de firewall la plus efficace se résume en 3 phrases:

- Initialisation du firewall.
- Par défaut, le firewall interdit toutes les connexions.
- On autorise uniquement les connexions strictement nécessaires. Cela signifie tout particulièrement que les règles acceptant les paquets devront être les plus "rigides" et limitatives possibles.

Sous Linux, la configuration du firewall va consister à lancer la commande "iptables" autant de fois que nécessaire pour construire une à une les différentes règles dont nous aurons besoin. Voyons maintenant comment construire un tel script.

Illustration : Principes généraux de filtrage

Exercice : Les opérations à faire ci-dessous sont nécessaires à la bonne poursuite de l'atelier.

- Avec votre utilisateur normal, créer dans un répertoire "/tmp/atelier_netfilter/". Par la suite, nous travaillerons exclusivement dans ce répertoire .
- Télécharger le script " restor_netfilter.sh", le sauver dans votre "/tmp/atelier_netfilter/".
- Donner les droits d'exécution à ce script : "chmod +x /tmp/atelier_netfilter/restor_netfilter.sh"
- Lancer un shell Linux (xterm, konsole, rxvt, etc...), passer en temps que root ("su -"), puis renter dans le "/tmp/atelier_netfilter/".

A partir de maintenant, vous devrez lancer tous les scripts de configuration de Netfilter que nous allons créer avec ce shell possédant les droits root. Attention de ne pas faire d'erreurs de manipulation !

II-2 Initialisation des chaînes

La première chose à faire pour construite notre script de firewall sera d'initialiser le firewall. Comme le script que nous allons écrire ne sait à priori pas dans quel état se trouve Netfilter (si des règles n'ont pas été déjà définies par exemple), il nous faudra commencer par supprimer toutes les précédentes règles (pré-définies et utilisateur). Et ce, pour toutes les tables de Netfilter.

Illustration : Initialisation des chaînes

Exercice : Écrire un script d'initialisation de Netfilter. On appellera ce script "initialisation1.sh"

Solution : La solution est téléchargeable ici.

L'ordre d'initialisation des tables n'a pas d'importance ici.

II-3 Cibles/politique de filtrage par défaut

Comme vu précédemment , il est nécessaire de configurer Netfilter afin de tout interdire par défaut.

La logique voudrait donc que l'on place à "DROP" la cible par défaut (ou politique par défaut) de chaque chaîne. Et ce, quelque soit la table. Cependant, comme nous désirons avant tout filtrer nos connexions, seul les cibles par défaut de la table FILTER sont à restreindre. Pour les tables NAT et MANGLE, il n'est donc pas nécessaire de mettre à DROP les cibles par défaut. Même si c'est techniquement faisable de restreindre les cibles par défaut des autres tables, cela alourdit considérablement le script de firewall, et entraîne une chute des performances réseau.

Illustration : Cibles/politique de filtrage par défaut

Exercice : Re-écrire le script d'initialisation vu ci-dessus, mais en rajoutant maintenant les politiques de filtrage par défaut. On appellera ce script "initialisation2.sh".

Améliorations : Penser à optimiser au mieux la sécurité, en plaçant les restrictions dans l'ordre de priorité décroissante.

Solution : La solution est téléchargeable ici.

A partir de maintenant, nous pourrons utiliser ce script comme base de travail pour nos autres scripts.

II-4 Filtrage de ports

Où en sommes nous ? Nous avons actuellement verrouillé notre Linux, en filtrant systématiquement ses communications réseaux.

Illustration : Filtrage de ports 1/2

Exercice :

- Utiliser la commande "ping" afin de tester le comportement de votre machine avec votre interface virtuelle "localhost". Puis refaire le test avec votre propre adresse IP réseau (la commande "ifconfig eth0" vous donnera votre adresse IP). Et enfin, faites le test sur la machine du présentateur.
- Rafraîchir la présente page HTML, en cliquant sur le bouton "Reload", combiné ou non avec la touche "shift".
  Remarque : Si le curseur de votre souris se transforme en une petite horloge, cela est tout à fait normal. Cela indique que votre navigateur est bloqué par le firewall....
- Lancer le script "restor_netfilter.sh", et refaire les précédents tests. Observer le résultat.

Bien, notre machine étant complètement verrouillée, nous allons voir maintenant comment lui laisser un peu communiquer avec les autres machines mises en réseau.

Illustration : Filtrage de ports 2/2

Exercice :

- Faire une copie du script "initialisation2.sh" en "filter1.sh".
- Écrire des règles de filtrage autorisant :
  - Toutes les connexions, sans restriction aucune, avec l'interface loopback ("lo").
  - Toutes les commandes de ping entrants et sortants par l'interface réseau "eth0". Rappel : La commande "ping" utilise le protocole "icmp".
  - La machine à accéder au serveur HTTP du présentateur. Le serveur HTTP tourne sur le port TCP/80.
- Vérifier la configuration de Netfilter :
  - Vous devez pouvoir "pinger" votre propre machine : "ping localhost".
  - Vous devez pouvoir "pinger" et recevoir les ping des autres machines du réseau : "ping adresse_ip_autre_machine".
  - Vous ne devez pas pouvoir utiliser la commande "nmap adresse_ip_autre_machine -p 80". Astuce : Utilisez "iptables -L -n -v" afin d'afficher les statistiques des paquets qui sont DROPés .
  - Vous devez pouvoir rafraîchir la présente page HTML.

Améliorations : Recopier le script "filter1.sh" en "filter1a.sh". Dans ce nouveau script, restreindre encore plus le firewall en rajoutant au filtrage du serveur HTTP du présentateur, le filtrage par votre propre adresses IP.

Solution : La solution est téléchargeable ici.

- La commande "iptables -L -n" vous donnera les informations relatives à vos règles de filtrages.
- Et la commande "iptables -L -n -v" vous donnera plus d'informations de statistiques. Pensez à élargir au maximum votre xterm pour afficher l'information de la manière la plus agréable et efficace possible.

II-5 Log

Maintenant nous allons travailler un peu avec les logs. Netfilter peut stocker ("logger" ou plus français "noter") des informations sur l'état du firewall, via le démon "syslogd". Ceci est particulièrement intéressant afin de rechercher une activité réseau suspecte (tentative d'intrusion), ou de rechercher la cause d'un mauvais fonctionnement du firewall

Illustration : Log

Exercice :.

- Recopier le script "filter1.sh" en "log1.sh".
- Y rajouter le log des tables INPUT et OUTPUT sur l'interface réseau eth0.

Améliorations : Recopier le script "log1.sh" en "log1a.sh". Rajouter un préfixe afin de différencier les logs entrants et sortants.

Solution : La solution est téléchargeable ici.

Lancer un xterm en pleine largeur d'écran, avec si possible une police assez petite. Puis taper, en temps que root, la commande "tail -f /var/log/messages".

Ou si possible (présence du paquetage "rxvt" sur la machine), télécharger rxvt_log depuis la machine du présentateur et en temps que root, taper "rxvt_log messages".

III Filtrages plus évolues

Dans cette partie, nous allons étudier les défauts du filtrage part ports vu précédemment . Puis, nous mettrons en pratiques des techniques de filtrages plus fines et plus efficaces.

III-1 Insuffisances du filtrage par ports

Nous allons utiliser "nmap", le scanneur de ports, afin de montrer les lacunes du filtrage par port.

- Le paramètre "-g Numéro_de_port" est tout particulièrement intéressant, car il permet d'initialiser des connexions en précisant le port source du scan.
- Le paramètre "-P0" permet à nmap de lancer son scan "en aveugle", sans vérifier auparavant si la machine cible est accessible.

Illustration : Insuffisances du filtrage par ports

Exercice :

- Éditer le script "filter1.sh" écrit précédemment ou la solution "filter1a.sh". Dans ce script, en retirer la protection par adresses IP (paramètres "-s" et "-d").
- Utiliser "nmap" en se faisant passer pour un serveur HTTP (paramètres "-g Numéro_de_port" et "-P0"), afin lancer un port scanning sur une autre machine du réseau.
- Conclusions ?

III-2 Suivi de connexion

La technique du suivi de connexion ("conntrack") rend inefficace la précédente astuce de port scanning. Elle permet aussi une sécurité très largement accrue, en restreignant au maximum les flux de connexions.

Le suivi de connexion se résume par les points suivants :

- Le module "-m state", à passer en paramètre à "iptables".
- Les états "--state Etat", à passer en paramètre à "iptables", après le "-m state". Les états possibles sont :
  - INVALID : Le paquet n'est associé avec aucune connexion connue.
  - NEW : Le paquet est une nouvelle connexion.
  - ESTABLISHED : Le paquet vient en réponse d'une connexion précédemment établie par la machine.
  - RELATED : Le paquet est en relation avec une connexion précédemment établie par la machine (exemple typique : les réponses depuis le port DATA d'un serveur FTP).

- Les modules kernels associés au suivi de connexion : ip_conntrack, ip_conntrack_ftp, ip_conntrack_proto_gre, ip_conntrack_h323, ip_conntrack_irc, ip_conntrack_pptp, etc...

Illustration : Suivi de connexion

Exercice : On reprend la philosophie du 1er exercice de filtrage, mais on va rendre notre firewall plus restrictif. Plus particulièrement, on utilisera la technique du suivit de connexion vu ici.

- Faire une copie du script "initialisation2.sh" en "filter2.sh".
- Écrire des règles de filtrage autorisant :
  - Toutes les connexions, sans restriction aucune, avec l'interface loopback ("lo").
  - Toutes les commandes de ping sortants (et uniquement sortants) par l'interface réseau "eth0".
    Rappel : La commande "ping" utilise le protocole "icmp".
  - La machine à accéder au serveur HTTP de n'importe quelle autre machine. Ceux-ci tournent sur les ports TCP/80.
- Vérifier la configuration de Netfilter :
  - Vous devez pouvoir "pinger" votre propre machine : "ping localhost".
  - Vous devez pouvoir "pinger" la machine du présentateur :
    "ping adresse_ip_présentateur ".
  - Les autres machines du réseau ne doivent pas pouvoir vous "pinger".
  - Vous ne devez pas pouvoir utiliser la commande
    "nmap adresse_ip_autre_machine -g 80 -p 80 -P0" (les autres machines se protègent).
  - Vous devez pouvoir rafraîchir la présente page HTML.
  - Le présentateur ne doit pas pouvoir accéder votre machine ("ping" et HTTP).

Améliorations : Recopier le script "filter2.sh" en "filter2a.sh". Dans ce nouveau script, rajouter les règles nécessaires afin que vous puissiez accéder aux serveurs FTP (port 21) de n'importe quelle autre machine. Attention, il y a un gros piège...

Solution : La solution est téléchargeable ici.

III-3 ULog

Les logs du firewall sont intéressants (voir même essentiels), mais il est dommage que l'on ne puisse pas les séparer des autres logs kernel. En effet, il arrive que le flots de logs de firewall soit très importants en volume, ce qui rend moins évident la lecture des autres logs du kernel (qui sont tout aussi important, si ce n'est plus).

ULOG est une solution très intéressante à ce problème. Il se décompose en :

- Un module du kernel (ipt_ULOG), gérant la cible "-j ULOG" d'"iptables", et chargé de générer les logs des connexions. Ce module est utilisable pour les kernels >= 2.4.18-pre8.
- Le démon " ulogd", travaillant de manière similaire à "syslogd", chargé de réceptionner et de stocker les logs de Netfilter
- Des modules à "ulogd", permettant de stocker les logs sur différents types de supports :
  - ulogd_LOGEMU : Sauve les logs dans fichier texte, à la manière de ce que fait "syslogd".
  - ulogd_OPRINT : Sauve le paquet intercepté complet dans un fichier binaire.
  - ulogd_MYSQL : Stock les logs dans une base de données MySQL.
  - ulogd_PGSQL : Stock les logs dans une base de données PostgreSQL.

Illustration : ULOG

Exercice : Mise en oeuvre de ULOG sur notre machine.

- Installer, configurer et démarrer le démon ULOG.
- Recopier le script "filter2.sh" en "log2.sh".
- Y rajouter le log des tables INPUT et OUTPUT sur l'interface réseau eth0, en utilisant la technique du ULOG.

Améliorations : Recopier le script "log2.sh" en "log2a.sh". Rajouter un préfixe afin de différencier les logs entrants et sortants.

Solution : La solution est téléchargeable ici.

Lancer un xterm en pleine largeur d'écran, avec si possible une police assez petite. Puis taper, en temps que root, la commande "tail -f /var/log/ulogd.syslogemu".

Ou si possible (présence du paquetage "rxvt" sur la machine), télécharger rxvt_log depuis la machine du présentateur et en temps que root, taper "rxvt_log netfilter".

III-4 Chaînes utilisateurs

Les chaînes utilisateurs ont le même rôle que les chaînes pré-défines, à savoir : Un conteneur à règles. L'intérêt des chaînes utilisateur n'est pas évident aux premiers abords, mais au final, elles sont très utiles.

Illustration : Chaînes utilisateurs 1/2

Exercice : Mise en oeuvre d'une simple chaîne utilisateur.

- Faire une copie du script "filter1.sh" en "userchaine1.sh".
- Passer toutes les politiques par défaut à ACCEPT.
- Simplifier les règles en supprimant les critères "-s" et "-d" (filtrage par adresses IP).
- Créer une chaîne utilisateur du nom de "LogDrop".
- Utiliser cette chaîne pour enregistrer et supprimer toutes les trames ne correspondant à aucune autres règles de filtrage.

Solution : La solution est téléchargeable ici.

Le petit exercice précédent n'est pas très spectaculaire. Mais il est nécessaire afin de bien comprendre l'exercice suivant, qui sera bien plus intéressant.

Dans ce qui va suivre, nous allons considérer que nous voulons avoir des liens privilégiés avec :

- "MACHINE 0", la machine du présentateur.
- "MACHINE 1" et "MACHINE 2", 2 machines de notre réseau local.

Nous ne désirons communiquer qu'avec ces trois machines, toutes les paquets IP a destination ou partant vers les autres machines devront être détruits.

Illustration : Chaînes utilisateurs 2/2

Exercice : Télécharger ce script utilisant la méthode classique de filtrage. A partir de ce script, il faudra :

- En faire une copie en "userchaine2b.sh".
- Modifier ce script afin de l'optimiser en utilisant la technique des chaînes utilisateurs. Indication : On peut considérer les "MACHINES 1 2 et 3" comme étant 3 interfaces réseaux différentes, mais sur lesquelles doivent être effectuer les mêmes contrôles.

Solution : La solution est téléchargeable ici.

- La solution est elle optimum ?
- Que faut il faire si l'on veut interdire les connexions à notre serveur HTTP ? La nouvelle solution est elle optimum à ce moment là ?

III-5 Contraintes de l'état NEW

Le problème de l'état NEW du suivi de connexion vient dans son mode de détection de ce qu'il considère comme étant une nouvelle connexion. Contrairement à ce que l'on peut penser, il ne se base tout simplement pas sur une handshake, ce qui peut l'amener à laisser des paquets se faisant passer pour une nouvelle connexion.

Illustration : Contraintes de l'état NEW 1/2

Si on ne veut pas utiliser l'état NEW, la technique de filtrage (pour des connexions initiées par notre machine) à utiliser peut être ceci :

- Interdiction des paquets non valide ("-state INVALID").
- Autorise les paquets (sortants ou entrants) de connexion déjà existantes ("--state ESTABLISHED,RELATED").
- Autorise les paquets sortants que nous avons initiés. C'est à dire, ceux utilisant le flag SYN ("--syn").

Illustration : Contraintes de l'état NEW 2/2

Exercice :

- Faire une copie du script "initialisation2.sh" en "nonew1.sh".
- Écrire des règles de filtrage autorisant :
  - Toutes les connexions, sans restriction aucune, avec l'interface loopback ("lo").
  - Les connexions "ping", HTTP et FTP vers l'extérieur ("eth0"), initiés par notre machine. Les paquets en retour de ces connexions doivent aussi être autorisé. On utilisera la technique vu ci-dessus.
  - Log tout le reste.

Solution : La solution est téléchargeable ici.

Ce script peut vous servir d'exemple pour votre propre firewall. On notera qu'il utilise un nombre de règles minimales. Et que nous contrôlons parfaitement les protocoles que nous voulons laisser sortir.

IV Firewall et serveurs

Mettre en place un démon (logiciel de type "serveur") n'est pas en soit très compliqué. Mais le configurer de telle manière à ce qu'il ne soit pas piraté n'est pas aisé. Le firewall peut assurer une partie de sa sécurité , mais la plus grande partie sera la responsabilité du démon lui même , comme par exemple "Apache" dans le cas d'un serveur HTTP.

Un firewall ne peut pas assurer à lui seul la sécurité d'un démon / serveur publique ! La configuration du démon est l'élément le plus important !

IV-1 Serveur HTTP

Nous n'allons pas nous intéresser ici à la configuration du démon HTTP lui-même. Cela ferait l'objet d'un atelier complet... Mais seulement de ce que peut faire Netfilter afin de protéger un tel démon.

Illustration : Serveur HTTP (web)

Exercice :

- Faire une copie du script "nonew1.sh" en "serveur1.sh".
- Écrire des règles de filtrage autorisant :
  - Les mêmes flux que pour l'exercice précédent .
  - Les autres machines du réseau à se connecter à notre serveur HTTP.

Améliorations : Recopier le script "serveur1.sh" en "serveur1a.sh". Dans ce nouveau script, autoriser les autres machines du réseau à "pinger" notre machine, mais en limitant ce flux à 1 ping/s (module "limit").

Solution : La solution est téléchargeable ici.

IV-2 Serveur FTP

Maintenant nous allons voir comment transcrire le travail précédent afin de protéger un serveur FTP.

Illustration : Serveur FTP (fichier)

Exercice :

- Faire une copie du script "serveur1.sh" en "serveur2.sh".
- Changer les règles du serveur HTTP et les adapter à un serveur FTP

Solution : La solution est téléchargeable ici.

Écouter les explications sur l'intérêt du module "ip_conntrack_ftp".

Conclusion

Dans cet atelier, nous avons abordé les principales techniques de Netfilter en terme de filtrage. Cependant, cet atelier ne peut résumer qu'une partie seulement des fonctionnalités de Netfilter. Aussi, vous êtes encouragé à fouiller la documentation que j'ai précédemment rédigé , les guides officiels de netfilter, le "man iptables", et bien évidement Google ou tout autre moteur de recherche, afin d'approfondir ce sujet.

Si vous devez retenir qu'une seul chose de cet atelier, c'est le principe général du filtrage : Initialisation du firewall / tout interdire par défaut / autoriser le strict minimum.

J'espère que cet atelier vous a intéressé, et que vous pourrez utiliser les notions abordées afin de configurer efficacement votre propre firewall Linux.

Si vous avez des remarques à faire sur cet atelier, des corrections à proposer ou des points à éclaircir, n'hésitez pas à me contacter (olivieraj@free.fr).

Remerciements

Je remercie le CUEFA (Grenoble, Isère / 38) et en particulière Romain KOBYLANSKI pour nous avoir fournit la salle et le matériel de cet atelier. Merci aussi à l'association Guilde pour l'organisation de l'atelier , et notamment (par ordre alphabétique) Edgar BONET, Christophe FIXOT et Jérôme KIEFFER.

License

supports aux conditions suivantes:

- tout lecteur ou utilisateur de ce document reconnaît avoir pris connaissance de ce qu'aucune garantie n'est donnée quant à son contenu, à tout point de vue, notamment véracité, précision et adéquation pour toute utilisation;
- il n'est procédé à aucune modification autre que cosmétique, changement de format de représentation, traduction, correction d'une erreur de syntaxe évidente, ou en accord avec les clauses ci-dessous;
- des commentaires ou additions peuvent êtres insérés à condition d'apparaître clairement comme tels; les traductions ou fragments doivent faire clairement référence à une copie originale complète, si possible à une copie facilement accessible;
- les traductions et les commentaires ou ajouts insérés doivent être datés et leur(s) auteur(s) doi(ven)t être identifiable(s) (éventuellement au travers d'un alias);
- cette licence est préservée et s'applique à l'ensemble du document et des modifications et ajouts éventuels (sauf en cas de citation courte), quelqu'en soit le format de représentation;
- quel que soit le mode de stockage, reproduction ou diffusion, toute personne ayant accès à une version numérisée ce document doit pouvoir en faire une copie numérisée dans un format directement utilisable et si possible éditable, suivant les standards publics, et publiquement documentés, en usage;
- la transmission de ce document à un tiers se fait avec transmission de cette licence, sans modification, et en particulier sans addition de clause ou contrainte nouvelle, explicite ou implicite, liée ou non à cette transmission. En particulier, en cas d'inclusion dans une base de données ou une collection, le propriétaire ou l'exploitant de la base ou de la collection s'interdit tout droit de regard lié à ce stockage et concernant l'utilisation qui pourrait être faite du document après extraction de la base ou de la collection, seul ou en relation avec d'autres documents.

Toute incompatibilité des clauses ci-dessus avec des dispositions ou contraintes légales, contractuelles ou judiciaires implique une limitation correspondante du droit de lecture, utilisation ou redistribution verbatim ou modifiée du document.

Olivier Allard-Jacquin

Site de référence : http://olivieraj.free.fr/ Last modified: Tue Apr 20 23:08:00 CEST 2004