TP Administration : Pare-feu

TP Administration

L'objectif de cette activité est de réaliser la mise en œuvre du pare-feu (*firewall*) UFW.

Mise en situation

Vous devez disposer d'un PC possédant un système d'exploitation Linux ou Windows et du logiciel de virtualisation *VirtualBox*. Le système invité sera une installation du **serveur Ubuntu 18.04 LTS**.

UFW (*Uncomplicated FireWall*)

`UFW` (*Uncomplicated FireWall*) est un nouvel outil de configuration simplifié en ligne de commande de Netfilter, qui donne une alternative à l'outil `iptables`. Il existe aussi `Gufw` qui est une interface graphique pour `ufw`.

Documentation : https://doc.ubuntu-fr.org/ufw

Uncomplicated Firewall est pré-installé sous Ubuntu, mais si besoin il faudrait installer le paquet `ufw`.

Question 1. Vérifier l'état de UFW.

```
$ sudo ufw status verbose
Status: inactive
```

UFW n'est pas activé par défaut, il faut donc l'activer avec la commande `ufw` :

```
USAGE: ufw [--dry-run] enable|disable|reload
```

Question 2. Activer UFW. Le pare-feu sera actif et lancé automatiquement au démarrage du système.

Question 3. Vérifier maintenant l'état de UFW. Quelle est la politique de filtrage par défaut ?

```
$ systemctl status ufw
ufw.service - Uncomplicated firewall
   Loaded: loaded (/lib/systemd/system/ufw.service; enabled; vendor preset: enabled)
   Active: active (exited) since Mon 2019-09-09 09:42:49 UTC; 2 days ago
     Docs: man:ufw(8)
 Main PID: 391 (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 1109)
   CGroup: /system.slice/ufw.service
...

$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

Les profils d'applications

Différentes applications peuvent enregistrer leurs profils auprès de UFW lors de leur installation. Ces profils permettent à UFW de gérer ces applications par leur nom. Par exemple, OpenSSH a un profil enregistré auprès de UFW.

Il est possible de lister ces applications :

```
$ sudo ufw app list
Available applications:
  OpenSSH
```

Et si vous avez installé le serveur HTTP Apache :

```
$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

Trois profils sont disponibles pour Apache :

- Apache : n'ouvre que le port 80 (trafic Web normal et non chiffré)
- Apache Full : ouvre le port 80 (trafic Web normal non crypté) et le port 443 (trafic crypté TLS/SSL)
- Apache Secure : n'ouvre que le port 443 (trafic crypté TLS/SSL)

Question 4. Autoriser les connexions SSH. Vérifier l'état de UFW. Tester.

Il faut s'assurer que le pare-feu autorise les connexions SSH afin de pouvoir se reconnecter la prochaine fois :

```
$ sudo ufw allow OpenSSH
Rule added
Rule added (v6)

$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Les règles par défaut

Il est possible d'utiliser les règles par défaut :

```
USAGE: ufw default allow|deny|reject [incoming|outgoing]
```

De manière générale, on distingue deux types de politique de sécurité :

- politique permissive (*open config*) : Cette politique repose sur le principe que par défaut on laisse tout passer puis on va restreindre pas à pas les accès et les services mais la sécurité risque d'avoir des failles.

— politique stricte (*close config*) : Cette politique repose sur le principe inverse. On commence par tout interdire, puis on décide de laisser seulement passer les services ou adresses désirés ou indispensables. La sécurité sera meilleure mais le travail sera plus difficile et cela peut même bloquer plus longtemps que prévu les utilisateurs. C'est évidemment la politique conseillée pour un pare-feu.

Pour une politique « **close config** » :

```
$ sudo ufw default deny incoming
La stratégie par défaut pour le sens « incoming » a été remplacée par « deny »
(veillez à mettre à jour vos règles en conséquence)

$ sudo ufw default deny outgoing
La stratégie par défaut pour le sens « outgoing » a été remplacée par « deny »
(veillez à mettre à jour vos règles en conséquence)

$ sudo ufw status verbose
État : actif
Journalisation : on (low)
Par défaut : deny (entrant), deny (sortant)
```

Pour une politique « **open config** » :

```
$ sudo ufw default allow incoming
La stratégie par défaut pour le sens « incoming » a été remplacée par « allow »
(veillez à mettre à jour vos règles en conséquence)

$ sudo ufw default allow outgoing
La stratégie par défaut pour le sens « outgoing » a été remplacée par « allow »
(veillez à mettre à jour vos règles en conséquence)

$ sudo ufw status verbose
État : actif
Journalisation : on (low)
Par défaut : allow (entrant), allow (sortant)
```

La journalisation

Et pour gérer la journalisation :

```
// Activer la journalisation :
$ sudo ufw logging on
Journalisation activée

// Désactiver la journalisation :
$ sudo ufw logging off
Journalisation désactivée
```

Exemples

L'ordre de déclaration des règles est très important, le système utilisant une politique « premier arrivé, premier servi ». Prenez donc soin d'ajouter vos règles spécifiques avant les règles générales lorsqu'elles concernent des éléments communs.

```
ufw [insert NUM] allow|deny|reject|limit [in|out] [log|log-all] PORT[/protocol]
```

Les exemples ci-dessous montrent l'utilisation de règles simples, par défaut les règles s'appliquent sur le trafic entrant (***in****coming*) :

— Ouverture du port 22 en TCP et UDP :

```
$ sudo ufw allow 22
```

— Ouverture du port 22 en TCP uniquement :

```
$ sudo ufw allow 22/tcp
```

— Ouverture du service ssh (port 22 en TCP et UDP) :

```
$ sudo ufw allow ssh
```

— Autorise les requêtes DNS (port 53 en TCP et UDP) en sortie :

```
$ sudo ufw allow out domain
```

Visualisation des règles et de leurs numéros :

```
$ sudo ufw status numbered
État : actif

     Vers                       Action     Depuis
     ----                       ------     ------
[ 1] 22                         ALLOW IN   Anywhere
[ 2] 53                         ALLOW OUT  Anywhere (out)
[ 3] 22                         ALLOW IN   Anywhere (v6)
[ 4] 53                         ALLOW OUT  Anywhere (v6) (out)
```

Suppression de la règle 1 :

```
$ sudo ufw delete 1
Suppression de :
 allow 22
Exécuter l'opération (o|n) ? o
La règle a été supprimée

...
```

UFW regarde dans la liste de services connus (`/etc/services`) pour appliquer les règles standards associées à des services. Pour accéder au manuel : `man ufw`

Et quelques règles plus complexes :

— Autoriser les accès du réseau local 10.0.0.0 :

```
$ sudo ufw allow from 10.0.0.0/8
```

— Interdire les accès au port 5000 pour la machine 192.168.0.1

```
$ sudo ufw deny proto udp from 192.168.0.1 to any port 5000
```

— Ouverture du service ssh (port 22 en TCP et UDP) :

```
$ sudo ufw allow ssh
```

— Autoriser le trafic HTTP et HTTPS entrants sur l'interface `enp0s3` :

```
$ sudo ufw allow in on enp0s3 proto tcp from any to any port 80,443
```

—

Visualisation des règles et de leurs numéros :

```
$ sudo ufw status numbered
État : actif

     Vers                       Action     Depuis
     ----                       ------     ------
[ 1] Anywhere                   ALLOW IN   10.0.0.0/8
[ 2] 5000/udp                   DENY IN    192.168.0.1
[ 3] 80,443/tcp on enp0s3       ALLOW IN   Anywhere
[ 4] 80,443/tcp (v6) on enp0s3 ALLOW IN Anywhere (v6)
```

UFW applique des règles `iptables` par défaut lors de son lancement. Vous pouvez les consulter et les modifier directement dans le fichier `/etc/ufw/before.rules`. Par exemple, cela peut s'avérer utile si vous voulez interdire les requêtes de `ping` (`ICMP` *Echo Request*).

Travail demandé

Question 5. Mettre en place une politique *close config* par défaut. Afficher les règles de filtrage.

Question 6. Autoriser le service SSH. Afficher les règles de filtrage.

Question 7. Autoriser/Interdire seulement le trafic local du réseau `192.168.52.0/24`. Tester. Afficher les règles de filtrage.

Question 8. Autoriser maintenant les connexions vers le port TCP 5001. Tester avec `netcat`. Afficher les règles de filtrage.

Question 9. Supprimer la règle autorisant les connexions vers le port TCP 5001. Tester avec `netcat`. Afficher les règles de filtrage.

Question 10. Interdire la possibilité de « pinger » votre serveur.

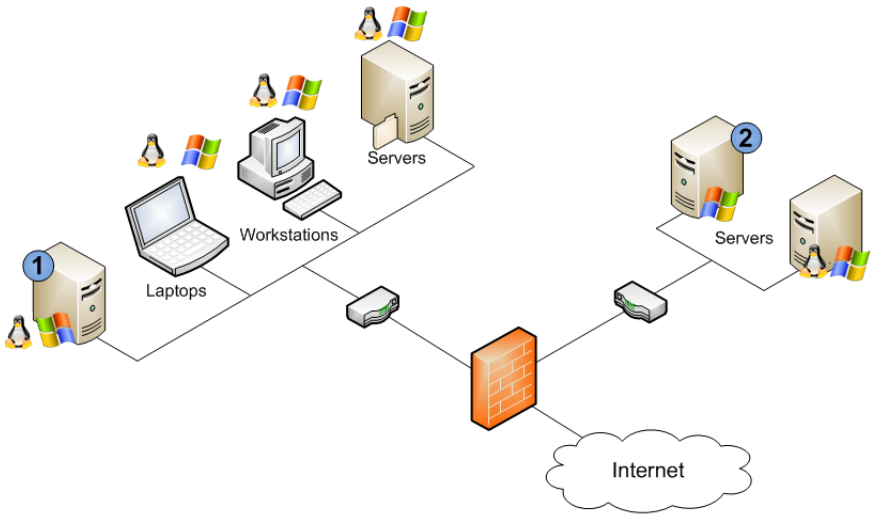
Annexes

Architecture réseau

Il existe plusieurs zones de sécurité commune aux réseaux. Ces zones déterminent un niveau de sécurité en fonction des accès réseaux et donnent les bases de l'architecture.

On considère en général trois zones ou réseaux :

— **réseaux externes** (*extranet*) ou « **publics** » : C'est le réseau généralement le plus ouvert (Internet par exemple). L'entreprise n'a pas ou très peu de contrôle sur les informations, les systèmes et les équipements qui se trouvent dans ce domaine.
— **réseaux internes** (*intranet*) ou « **privés** » : Les éléments de ce réseau doivent être sérieusement protégés. C'est souvent dans cette zone que l'on trouve les mesures de sécurité les plus restrictives et c'est donc le réseau le moins ouvert.
— **réseaux intermédiaires** (*dmz*) : Cette zone est un compromis entre les deux précédentes. Ce réseau est composé de services fournis aux réseaux internes et externes. Les services publiquement accessibles (serveurs de messagerie, Web, FTP et DNS le plus souvent) sont destinés aux utilisateurs internes et aux utilisateurs par Internet. Cette zone, appelée réseau de services ou de **zone démilitarisée DMZ** (*De-Militarized Zone*), est considérée comme la zone moins protégée de tout le réseau.



Pare-feu (*firewall*)

Un système pare-feu (*firewall*) est un dispositif conçu pour examiner et éventuellement bloquer les échanges de données entre réseaux.

Le pare-feu joue le rôle de filtre et peut donc intervenir à plusieurs niveaux du modèle à couches. De manière générale, le pare-feu va analyser les protocoles utilisés dans une communication afin de chercher des correspondances avec les règles définies et décider de l'action à appliquer (accpeter ou rejeter par exemple).

Il existe trois types principaux de pare-feu :

— filtrage de paquets (*firewall stateless*)
— filtrage de paquets avec état (*firewall stateful*)
— *proxy* applicatif

Filtrage de paquets (*firewall stateless*)

Les pare-feu de filtrage de paquets sont généralement des routeurs qui permettent d'accorder ou de refuser l'accès en fonctions des éléments suivants :

— l'adresse source, l'adresse destination
— le numéro de port
— le protocole

Les limites de cette technique :

— Manque de souplesse : Si cette approche offre une défense simple et efficace, elle manque de souplesse pour que sa mise en place en protection d'un réseau, dans la majorité des cas, ne s'avère pas bloquante pour le bon fonctionnement des systèmes du réseau concerné. Pour contourner ce défaut, l'administrateur est rapidement contraint à autoriser trop d'accès pour que son pare-feu offre encore la moindre protection réelle.
— Difficulté pour gérer certains protocoles : Certains protocoles sont particulièrement délicats à gérer à l'aide de cette technique comme FTP (le suivi des échanges FTP est une opération complexe) ou TCP (protocole de type connecté).

Filtrage de paquets avec état (*firewall stateful*)

La technologie *stateful* est l'une des deux réponses possibles aux limites du filtre de paquets.

Un *firewall stateful* inclut toutes les fonctionnalités d'un filtrage de paquet, auxquelles il ajoute la capacité de conserver la trace des sessions et des connexions dans des tables d'état interne. Tout échange de données est soumis à son approbation et adapte son comportement en fonction des états.

Cette technique convient aux protocoles de type connecté (TCP). Certains protocoles (UDP et ICMP) posent un problème supplémentaire : aucune notion de connexion n'y est associée. Le pare-feu est donc amené à examiner les paquets, et peut seulement gérer des *timeout*, souvent de l'ordre d'une minute.

Les limites de cette technique :

— Risque d'accès illimité : Si cette approche apporte une amélioration certaine par rapport à la technique du filtrage simple de paquets, elle se borne cependant à autoriser ou interdire l'accès à un service donné. Dès lors que l'accès à un service est accordé, celui-ci est illimité.
— Faille interne au *firewall* : D'autre part, un tel système ne protège aucunement un serveur contre les attaques s'appuyant sur des failles du logiciel utilisé pour fournir le service.

Néanmoins, le pare-feu de type *stateful* est considéré, par les administrateurs réseau, comme la technologie minimum pour une sécurisation.

Proxy applicatif

Les *firewalls* de type *proxy* applicatif (ou passerelle applicative) ont pour ambition de répondre aux problèmes soulevés par les *firewall stateless* et les *firewall stateful*.

Ces systèmes se substituent au serveur ou au client qu'ils ont pour mission de défendre pour :

— traiter les requêtes et réponses à la place du système à protéger,
— les transmettre, après d'éventuelles modifications
— ou les bloquer

Le pare-feu de ce type joue le rôle de canal et d'interpréteur en agissant aux niveaux des protocoles de la couche Application. Cette approche permet, en principe, d'atteindre le plus haut niveau de sécurité.

De nombreux *firewalls* de ce type sont disponible sur le marché, comme Raptor de Axent, Gauntlet de NAI, racheté par Secure Computing, Sidewinder de Secure Computing, et M-Wall de Matranet, pour ne citer que les plus connus.

Cette technologie, bien que prometteuse, est cependant confrontée à certaines difficultés.

Les limites de cette technique :

— Adapatabilité : En premier lieu, tout protocole transitant par le pare-feu doit être connu de celui- ci, pour pouvoir bénéficier de ses capacités de *proxy*, ce qui exclut l'usage d'applications et protocoles "maison", ou plus simplement peu répandus.
— Difficulté : En outre, la gamme de protocoles à examiner étant particulièrement vaste, il est extrêmement délicat d'obtenir un système éliminant réellement toutes les attaques envisageables, les développeurs de *firewalls* applicatifs ne pouvant que difficilement maîtriser tous ces protocoles.
— Failles du *proxy* : Un tel système a pour rôle, comme les clients et serveurs qu'il défend, d'interpréter, comprendre, et réécrire les requêtes et réponses qu'il reçoit. Or, c'est précisément ce type de fonctionnalités qui est à l'origine d'une vaste majorité des failles applicatives.
— Performance : Enfin, le gain de sécurité obtenu à l'aide du *proxy* applicatif se paie en termes de performances. Les tâches que ce système a pour rôle de réaliser étant nettement plus complexes que celles du *firewall stateful*, une machine puissante est nécessaire pour faire tourner ce type de logiciel, et il est rare que des débits réseau supérieurs au cinquième des débits gérés par un *firewall stateful* puissent être traités.

Autres possibilités des pare-feu

Les fabricants de pare-feu ont tendance à intégrer un maximum de fonctionnalités :

— filtrage de contenu (URL, *spam mails*, code ActiveX, applets Java, ...)
— réseau virtuel privé (VPN) : les VPN permettent de canaliser un trafic sécurisé d'un point à un autre sur des réseaux généralement hostile (Internet par exemple). Checkpoint et Cisco intègrent des services VPN à leur offres de pare-feu.
— la traduction d'adresse réseau NAT (*Network Address Translation*) : ce service permet de mettre en correspondance des adresses réservées ou illégales avec des adresses valides. Les premiers dispositifs NAT qui apparaissent en entreprise sont souvent des produits pare-feu.
— l'équilibrage de charge : va permettre de segmenter un trafic de façon répartie (orienter le trafic Web et FTP par exemple)
— la tolérance de pannes : certains pare-feu, comme CISCO/PIX ou Nokia/Checkpoint supportent ces fonctionnalités (généralement exécution des pare-feu par paire pour permettre une disponibilité élevée (*High-Availability*)
— la détection des intrusions (IDS)