

Teaching-HEIGVD-SRX-2019-Laboratoire-Firewall

ATTENTION : Commencez par créer un Fork de ce repo et travaillez sur votre fork.

Clonez le repo sur votre machine. Vous retrouverez notamment dans ce repo le fichier **Dockerfile** indispensable pour [l'ajout des conteneurs et configuration du réseau](#).

Vous pouvez répondre aux questions en modifiant directement votre clone du README.md ou avec un fichier pdf que vous pourrez uploader sur votre fork.

Le rendu consiste simplement à compléter toutes les parties marquées avec la mention "LIVRABLE". Le rendu doit se faire par une "pull request". Envoyer également le hash du dernier commit et votre username GitHub par email au professeur et à l'assistant

Table de matières

[Introduction](#)

[Echéance](#)

[Topologie](#)

[Adressage](#)

[Cahier des charges du réseau](#)

[Regles de filtrage](#)

[Installation de l'environnement virtualisé](#)

[Tests des connections et exemple de l'application d'une règle](#)

[Règles pour le protocole DNS](#)

[Règles pour les protocoles HTTP et HTTPS](#)

[Règles pour le protocole ssh](#)

[Règles finales iptables](#)

Introduction

L'objectif principal de ce laboratoire est de familiariser les étudiants avec les pare-feu et en particulier avec netfilter et iptables. En premier, une partie théorique permet d'approfondir la rédaction de règles de filtrage.

Par la suite, la mise en pratique d'un pare-feu permettra d'approfondir la configuration et l'utilisation d'un pare-feu ainsi que la compréhension des règles.

Auteurs

Ce texte se réfère au laboratoire « Pare-feu » à suivre dans le cadre du cours Sécurité des Réseaux, 2019, version 6.2. Au cours du temps, il a été rédigé, modifié et amélioré par les co-auteurs suivants : Gilles-Etienne Vallat, Alexandre Délez, Olivia Manz, Patrick Mast, Christian Buchs, Sylvain Pasini, Vincent Pezzi, Yohan Martini, Ioana Carlson, Abraham Rubinstein et Frédéric Saam.

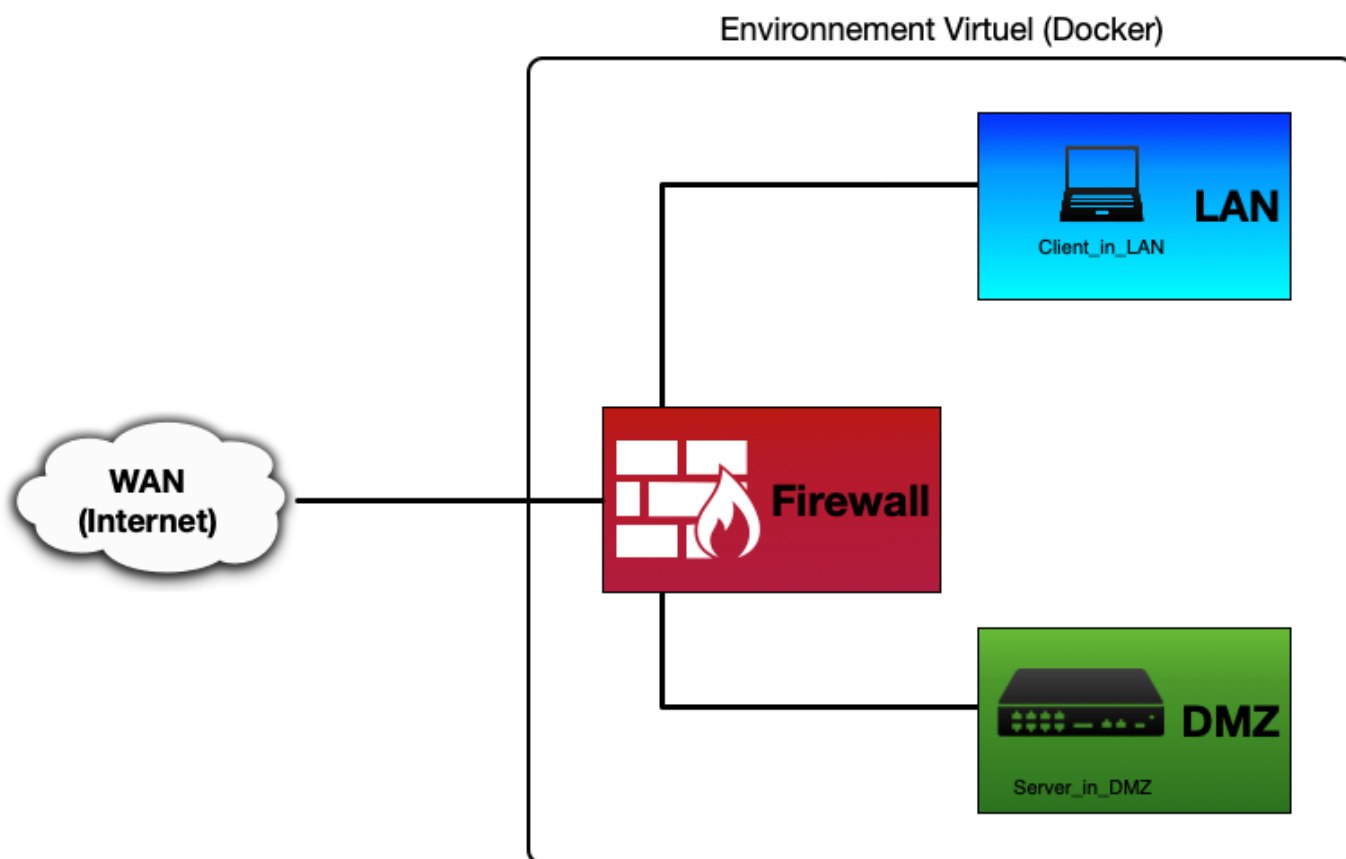
Echéance

Ce travail devra être rendu le dimanche après la fin de la 2ème séance de laboratoire, soit au plus tard, **le 31 mars 2019, à 23h.**

Réseaux cible

Topologie

Durant ce laboratoire, nous allons utiliser une seule topologie réseau :



Notre réseau local (LAN) sera connecté à Internet (WAN) au travers d'un pare-feu. Nous placerons un serveur Web en zone démilitarisée (DMZ).

Par conséquent, nous distinguons clairement trois sous-réseaux :

- Internet (WAN), le réseau de l'école servira de WAN,
- le réseau local (LAN),
- la zone démilitarisée (DMZ).

Ce réseau sera créé de manière virtuelle. Il sera simulé sur un seul ordinateur utilisant trois conteneurs Docker basés sur le système d'exploitation Ubuntu :

- La première machine, Firewall, fait office de pare-feu. Elle comporte trois interfaces réseaux. Afin que ce poste puisse servir de pare-feu dans notre réseau, iptables sera utilisé.
- La seconde machine, Client_In_LAN, fait office de client dans le réseau local (LAN).
- La dernière machine, Server_In_DMZ, fait office de serveur Web en (DMZ).

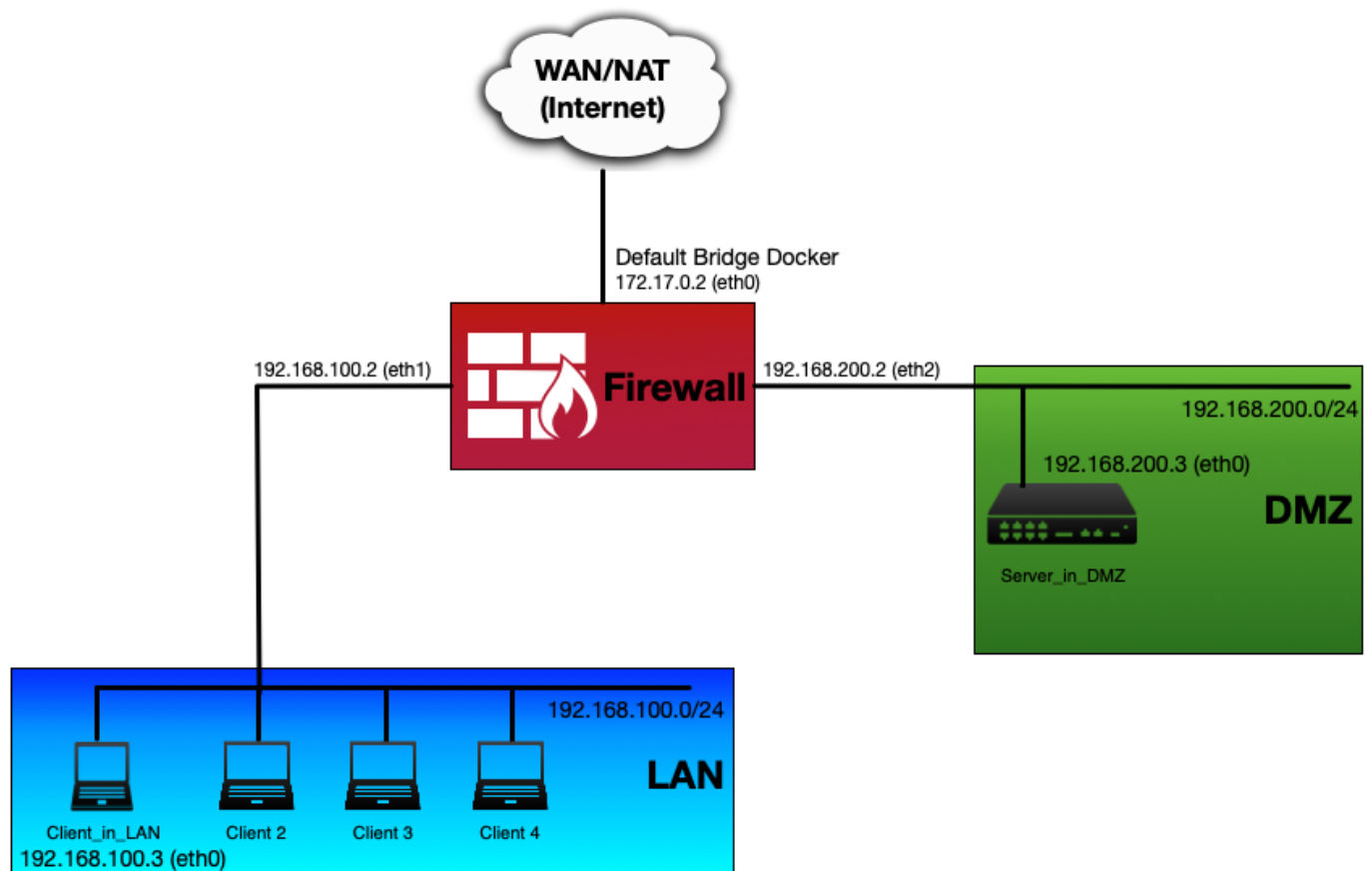
Nous allons utiliser les trois interfaces réseaux de la machine Firewall afin de pouvoir connecter le LAN et la DMZ à Internet (WAN). Les machines Client_In_LAN et Server_In_DMZ comportent chacune une interfaces réseau eth0.

Plan d'adressage

Afin de bien spécifier le réseau, il est nécessaire d'avoir un plan d'adressage précis. C'est la liste des réseaux que vous utiliserez, comprenant pour chaque interface l'adresse IP ainsi que le masque de sous-réseau. Pour ce laboratoire, nous vous imposons le plan d'adressage suivant :

- Le réseau "LAN" → 192.168.100.0/24
- Le réseau "DMZ" → 192.168.200.0/24
- Le réseau "WAN" sera défini par le NAT interne du réseau Docker

Les adresses IP sont définies dans le schéma ci-dessous :



Cahier des charges du réseau

Avant de configurer les règles, il est primordial de connaître les besoins de notre réseau. Ceci afin de laisser passer les flux légitimes lors de la rédaction des règles.

Le but du **LAN** est de fournir aux utilisateurs de votre réseau un accès à Internet ; à certains services de base uniquement en empêchant les connexions provenant de l'extérieur. Il faudra tout de même laisser entrer les

paquets répondants aux requêtes de notre LAN. Une seule machine est présente sur ce réseau. Il s'agit de la machine dont le nom est **Client_In_LAN**. (il est très facile de rajouter de machines supplémentaires sur le LAN utilisant Docker).

La **DMZ** est un réseau réservé aux serveurs que l'on veut rendre accessibles depuis l'extérieur et l'intérieur de notre réseau. Par exemple, si nous voulons publier un site web que l'on héberge, il faut accepter des connexions sur le serveur web; dans ce cas, nous ne pouvons pas le placer dans le LAN, cela constituerait un risque. Nous accepterons donc les connexions entrantes dans la DMZ, mais seulement pour les services que l'on désire offrir. Le serveur Web situé dans la DMZ est simulé par la machine **Server_In_DMZ**.

Le **WAN** n'est que l'accès à Internet. Il est connecté au réseau de l'école à travers le système de réseau fourni par Docker.

Pour établir la table de filtrage, voici les **conditions à respecter** dans le cadre de ce laboratoire :

1. Les **serveurs DNS** utilisés par les postes dans le LAN sont situés sur le WAN. Les services DNS utilisent les ports UDP 53 et TCP 53.
2. Laisser passer les **PING** uniquement du LAN au WAN, du LAN à la DMZ et de la DMZ au LAN pour les tests. Le ping utilise le protocole ICMP (echo request et echo reply).
3. Les clients du **LAN** doivent pouvoir ouvrir des connexions HTTP pour accéder au web. Le protocole HTTP utilise les ports TCP 80 et typiquement aussi le 8080.
4. Les clients du **LAN** doivent pouvoir ouvrir des connexions HTTPS pour accéder au web. Le protocole HTTPS utilise le port TCP 443.
5. Le serveur **web en DMZ** doit être atteignable par le WAN et le LAN et n'utilise que le port 80.
6. Le serveur de la DMZ peut être commandé à distance par **ssh** depuis votre client du LAN **uniquement**. Le service ssh utilise le port TCP 22.
7. Le firewall peut être configuré à distance par **ssh** depuis votre client du LAN **uniquement**.
8. **Toute autre action est par défaut interdite.**

Regles de filtrage

- a. En suivant la méthodologie vue en classe, établir la table de filtrage avec précision en spécifiant la source et la destination, le type de trafic (TCP/UDP/ICMP/any), les ports sources et destinations ainsi que l'action désirée (****Accept**** ou ****Drop****, éventuellement ****Reject****).

*Pour l'autorisation d'accès (**Accept**), il s'agit d'être le plus précis possible lors de la définition de la source et la destination : si l'accès ne concerne qu'une seule machine (ou un groupe), il faut préciser son adresse IP ou son nom (si vous ne pouvez pas encore la déterminer), et non la zone. Appliquer le principe inverse (être le plus large possible) lorsqu'il faut refuser (**Drop**) une connexion.*

Lors de la définition d'une zone, spécifier l'adresse du sous-réseau IP avec son masque (par exemple, "/24" correspond à 255.255.255.0) ou l'interface réseau (par exemple : "interface WAN") si l'adresse du sous-réseau ne peut pas être déterminé avec précision.

LIVRABLE : Remplir le tableau

Adresse IP source	Adresse IP destination	Type	Port src	Port dst	Action
-------------------	------------------------	------	----------	----------	--------

Adresse IP source	Adresse IP destination	Type	Port src	Port dst	Action
*	*	*	*	*	Drop
192.168.100.0/24	*	TCP	-	53	Accept
192.168.100.0/24	*	UDP	-	53	Accept
192.168.100.0/24	*	ICMP	-	echo-request	Accept
192.168.200.0/24	192.168.100.0/24	ICMP	-	echo-request	Accept
192.168.100.0/24	192.168.200.0/24	ICMP	-	echo-request	Accept
192.168.200.0/24	*	TCP	-	80	Accept
192.168.200.0/24	*	TCP	-	8080	Accept
192.168.100.0/24	*	TCP	-	443	Accept
192.168.100.0/24	192.168.200.0/24	TCP	-	80	Accept
*	192.168.200.0/24	TCP	-	80	Accept
192.168.100.3	192.168.200.3	TCP	-	22	Accept
192.168.100.3	192.168.100.2	TCP	-	22	Accept

- Doit on spécifier les règles de filtrage aller-retour ou une règle de filtrage est implicitement avec état et on ignore le retour ? (actuellement une règle allé et une règle retour)

Installation de l'environnement virtualisé

Ce chapitre indique comment installer l'environnement. Il se base sur des outils gratuits, téléchargeables sur Internet.

Matériel

Il est possible d'utiliser les mêmes instructions sur une version de Windows ou un système Linux ou Mac OS X.

Afin d'installer les différents logiciels présentés ici, il faut disposer d'un ordinateur (avec les droits administrateur).

Installation de Docker

Docker est un logiciel permettant de créer des conteneurs virtuels afin de simuler diverses configurations. Nous l'utiliserons pour exécuter les trois machines dont nous aurons besoin pour ce laboratoire. L'installation de Docker ne comporte pas de difficulté particulière. Une installation « par défaut » suffira. Il est possible d'utiliser une version que vous avez déjà installée ou une version téléchargée, mais la documentation pour ce laboratoire est fournie pour la version 2.0.0.3. Si vous rencontrez des problèmes, une mise à jour de Docker es peut-être la solution.

Vous pouvez installer Docker pour Windows et Mac OS depuis <https://www.docker.com/products/docker-desktop>

Pour Linux, referez-vous au gestionnaire de paquets de votre distribution.

Installation de Git

Vous avez probablement déjà installé Git pour d'autres cours ou projets. Si ce n'est pas le cas, vous pouvez prendre la bonne version pour votre OS depuis <https://git-scm.com/download/>

Ajout des conteneurs et configuration du réseau

Le repertoire [scripts](#) contient des scripts qui automatisent des parties de ce travail. Il es cependant conseillé de la faire manuellement pour mieux comprendre la procédure.

Nous allons commencer par créer les réseaux **LAN** et **DMZ** dans le système réseau de Docker. Il suffit de taper les commandes suivantes :

```
docker network create --subnet 192.168.100.0/24 lan
docker network create --subnet 192.168.200.0/24 dmz
```

Vous pouvez vérifier que les réseaux ont été créés avec `docker network ls`. Ils devraient se trouver dans la liste.

Les images utilisées pour les conteneurs sont basées sur l'image officielle Ubuntu. Le fichier [Dockerfile](#) que vous avez téléchargé contient les informations nécessaires pour la génération de l'image. On utilise la commande suivante pour la générer (**attention au point à la fin de la ligne !**) :

```
docker build -t labofirewall .
```

A partir de l'image, nous pouvons maintenant créer et executer les conteneurs. On va commencer avec le firewall :

```
docker run -itd --cap-add=NET_ADMIN --cap-add=NET_RAW --name firewall --
hostname Firewall labofirewall /bin/bash
```

Cette image est uniquement connectée au bridge par défaut de Docker. Nous allons maintenant connecter nous deux réseaux créés précédemment :

```
docker network connect lan firewall
docker network connect dmz firewall
```

On peut maintenant générer et lancer les conteneurs pour le serveur dans la DMZ et le client dans le réseau LAN:

```
docker run -itd --net lan --cap-add=NET_ADMIN --cap-add=NET_RAW --name lan  
--hostname Client_in_LAN labofirewall /bin/bash  
docker run -itd --net dmz --cap-add=NET_ADMIN --cap-add=NET_RAW --name dmz  
--hostname Server_in_DMZ labofirewall /bin/bash
```

Communication avec les VMs et configuration du firewall

Afin de simplifier vos manipulations, les VMs ont été configurées avec les noms suivants :

- firewall
- lan
- dmz

Pour accéder au terminal de l'une des machines, il suffit de taper :

```
docker exec -it <nom_de_la_machine> /bin/bash
```

Par exemple, pour ouvrir un terminal sur votre firewall :

```
docker exec -it firewall /bin/bash
```

Vous pouvez bien évidemment lancer des terminaux avec les trois machines en même temps !

Configuration de base

La plupart de paramètres sont déjà configurés correctement sur les trois machines. Il est pourtant nécessaire de rajouter quelques commandes afin de configurer correctement le réseau pour le labo.

Vous pouvez commencer par vérifier que le ping n'est pas possible actuellement entre les machines. Depuis votre Client_in_LAN, essayez de faire un ping sur le Server_in_DMZ (cela ne devrait pas fonctionner !) :

```
ping 192.168.200.3
```

LIVRABLE : capture d'écran de votre tentative de ping.

```
root@Client_in_LAN: /
File Edit View Search Terminal Help
root@Client_in_LAN:/# ping 192.168.200.3
PING 192.168.200.3 (192.168.200.3) 56(84) bytes of data.

^C
--- 192.168.200.3 ping statistics ---
157 packets transmitted, 0 received, 100% packet loss, time 159725ms
```

En effet, la communication entre les clients dans le LAN et les serveurs dans la DMZ doit passer à travers le Firewall. Il faut donc définir le Firewall comme passerelle par défaut pour le client dans le LAN et le serveur dans la DMZ.

Configuration du client LAN

Dans un terminal de votre client, taper les commandes suivantes :

```
ip route del default
ip route add default via 192.168.100.2
```

Configuration du serveur dans la DMZ

Dans un terminal de votre serveur, taper les commandes suivantes :

```
ip route del default
ip route add default via 192.168.200.2
```

La communication devrait maintenant être possible entre les deux machines. Faites un nouveau test de ping, cette fois-ci depuis le serveur vers le client :

```
ping 192.168.100.3
```

LIVRABLE : capture d'écran de votre nouvelle tentative de ping.

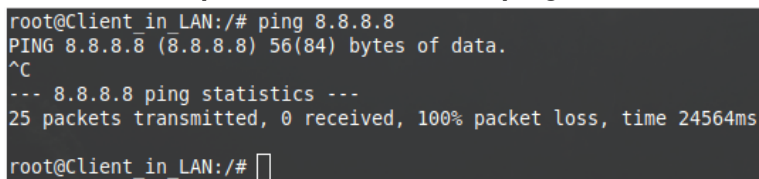
```
root@Server_in_DMZ:/# ping 192.168.100.3
PING 192.168.100.3 (192.168.100.3) 56(84) bytes of data.
64 bytes from 192.168.100.3: icmp_seq=1 ttl=64 time=0.279 ms
64 bytes from 192.168.100.3: icmp_seq=2 ttl=64 time=0.126 ms
64 bytes from 192.168.100.3: icmp_seq=3 ttl=64 time=0.132 ms
64 bytes from 192.168.100.3: icmp_seq=4 ttl=64 time=0.116 ms
64 bytes from 192.168.100.3: icmp_seq=5 ttl=64 time=0.123 ms
^C
--- 192.168.100.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.116/0.155/0.279/0.062 ms
root@Server_in_DMZ:/#
```

La communication est maintenant possible entre les deux machines. Pourtant, si vous essayez de communiquer depuis le client ou le serveur vers l'Internet, ça ne devrait pas encore fonctionner sans une manipulation supplémentaire au niveau du firewall. Vous pouvez le vérifier avec un ping depuis le client ou le serveur vers une adresse Internet.

Par exemple :

```
ping 8.8.8.8
```

LIVRABLE : capture d'écran de votre ping vers l'Internet.



```
root@Client_in_LAN:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
25 packets transmitted, 0 received, 100% packet loss, time 24564ms
root@Client_in_LAN:/#
```

Configuration réseau du firewall

Dans un terminal de votre firewall, vous pouvez utiliser l'éditeur **nano** déjà installé pour éditer le fichier `/etc/ssh/sshd_config`.

Il faudra changer la ligne :

```
#PermitRootLogin prohibit-password
```

à :

```
PermitRootLogin yes
```

et enregistrer et fermer le fichier en question.

Ensuite, il faut taper les commandes suivantes :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
service nginx start
service ssh start
```

La commande **iptables** définit une règle dans le tableau NAT qui permet la redirection de ports et donc, l'accès à l'Internet pour les deux autres machines.

Les deux autres commandes démarrent les services Web et SSH du serveur.

ATTENTION : Il faudra aussi définir un mot de passe pour les connexions ssh. Pour cela, utiliser la commande `passwd`.

- `toor`

Vérifiez que la connexion à l'Internet est maintenant possible depuis les deux autres machines. Pas besoin de capture d'écran.

Manipulations

Création de règles

Une règle permet d'autoriser ou d'interdire une connexion. `iptables` met à disposition plusieurs options pour la création de ces règles. En particulier, on peut définir les politiques par défaut « Policy », des règles de filtrage pour le firewall (tableau filter) ou des fonctionnalités de translation d'adresses (tableau nat) :

- Policy permet d'appliquer des règles générales (**vous devez configurer vos politiques en premier**)
- Le tableau filter permet d'appliquer des règles de filtrage propres d'un firewall
- Le tableau nat permet de paramétrer la translation d'adresses

`iptables` vous permet la configuration de pare-feux avec et sans état. **Pour ce laboratoire, vous allez utiliser le mode avec état.**

Chaque règle doit être tapée sur une ligne séparée. Référez-vous à la théorie et appuyez-vous sur des informations trouvées sur Internet pour traduire votre tableau de règles de filtrage en commandes `iptables`. Les règles prennent effet immédiatement après avoir appuyé sur <enter>. Vous pouvez donc les tester au fur et à mesure que vous les configurez.

Sauvegarde et récupération des règles

Important : Les règles de filtrage définies avec `iptables` ne sont pas persistantes (elles sont perdues après chaque redémarrage de la machine firewall). Pour sauvegarder votre configuration de firewall au fur et à mesure que vous avancez, vous pouvez utiliser les outils `iptables-save` et `iptables-restore`.

Sauvegarder la configuration du firewall dans le fichier `iptables.conf` :

```
iptables-save > iptables.conf
```

Récupérer la config sauvegardée :

```
iptables-restore < iptables.conf
```

→ Note : pour plus de détails, la commande `iptables -L` affiche toutes les règles en vigueur.

- Note : avant chaque installation, la commande `iptables -F` efface les règles en vigueur.
- Note : avant chaque installation, la commande `iptables -X` efface les chaînes.
- Note : Puisque vous travaillez depuis un terminal natif de votre machin hôte, vous pouvez facilement copier/coller les règles dans un fichier local. Vous pouvez ensuite les utiliser pour reconfigurer votre firewall en cas de besoin.

Tests des connections et exemple de l'application d'une règle

Pour chaque manipulation, il est important de **garder les règles déjà créées**, les nouvelles sont ajoutées aux existantes.

Pour commencer sur une base fonctionnelle, nous allons configurer le pare-feu pour accepter le **ping** dans certains cas. Cela va permettre de tester la connectivité du réseau.

Le but est de configurer les règles pour que le pare-feu accepte

- les ping depuis le LAN sur les machines de la DMZ,
- les ping depuis le LAN sur le WEB,
- les ping depuis la DMZ vers le LAN.

Ceci correspond a la **condition 2** du cahier des charges.

Commandes iptables :

```
docker="docker exec -t firewall"
# Not necessary for ICMP protocole but TCP and UDP need to be state full
# return traffic
echo "0. Allows return traffic : RELATED, ESTABLISHED"
$docker iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j
ACCEPT
$docker iptables -A FORWARD -m conntrack --ctstate INVALID -j DROP
# ssh return traffic for Clie_in_LAN
$docker iptables -A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j
ACCEPT

#-----
# 1. PING
#-----
echo "1. Allows pings echo-request & echo-reply for LAN"
# LAN --> WAN
$docker iptables -A FORWARD -s 192.168.100.0/24 -i eth2 -p icmp --icmp-type
8 -j ACCEPT
$docker iptables -A FORWARD -d 192.168.100.0/24 -i eth0 -p icmp --icmp-type
0 -j ACCEPT

# DMZ --> LAN
$docker iptables -A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -p icmp
--icmp-type 8 -j ACCEPT
```

```
$docker iptables -A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -p icmp
--icmp-type 0 -j ACCEPT

# LAN --> DMZ
$docker iptables -A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -p icmp
--icmp-type 8 -j ACCEPT
$docker iptables -A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -p icmp
--icmp-type 0 -j ACCEPT
```

Questions

- b. Afin de tester la connexion entre le client (Client_in_LAN) et le WAN, tapez la commande suivante depuis le client :

```
ping 8.8.8.8
```

Faire une capture du ping.

```
root@Client_in_LAN:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=120 time=9.85 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=120 time=10.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=120 time=9.69 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=120 time=10.3 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=120 time=9.79 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=120 time=9.85 ms
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 9.696/9.981/10.391/0.285 ms
root@Client_in_LAN:/#
```

- c. Testez ensuite toutes les règles, depuis le Client_in_LAN puis depuis le serveur Web (Server_in_DMZ) et remplir le tableau suivant :

De Client_in_LAN à	OK/KO	Commentaires et explications
Interface DMZ du FW	KO	LAN n'est pas autorisé à ping le firewall
Interface LAN du FW	KO	LAN n'est pas autorisé à ping le firewall
Client LAN	OK	LAN autorisé à ping lui même ou autre client LAN (ne passe pas par le firewall)
Serveur WAN	OK	LAN est autorisé ping les servers WAN

De Server_in_DMZ à	OK/KO	Commentaires et explications
Interface DMZ du FW	KO	DMZ n'est pas autorisé à ping le firewall
Interface LAN du FW	KO	DMZ n'est pas autorisé à ping le firewall
Serveur DMZ	OK	DMZ autorisé à ping lui même ou autre Serveur DMZ (ne passe pas par le firewall)
Serveur WAN	NOK	DMZ n'est pas autorisé à ping un Serveur DMZ

Règles pour le protocole DNS

- d. Si un ping est effectué sur un serveur externe en utilisant en argument un nom DNS, le client ne pourra pas le résoudre. Le démontrer à l'aide d'une capture, par exemple avec la commande suivante :

```
ping www.google.com
```

- Faire une capture du ping.

```
root@Client_in_LAN:/# ping www.google.com
ping: www.google.com: Temporary failure in name resolution
root@Client_in_LAN:/#
```

- Créer et appliquer la règle adéquate pour que la **condition 1 du cahier des charges** soit respectée.

Commandes iptables :

```
#-----
# 2. DNS
#-----
echo "2. Allows DNS lookups (tcp, udp port 53) for LAN"
# LAN --> WAN
$docker iptables -A FORWARD -s 192.168.100.0/24 -i eth2 -p udp --dport 53 -
j ACCEPT
$docker iptables -A FORWARD -s 192.168.100.0/24 -i eth2 -p tcp --dport 53 -
j ACCEPT
```

- e. Tester en réitérant la commande ping sur le serveur de test (Google ou autre) :

```
root@Client_in_LAN:/# ping www.google.ch
PING www.google.ch (172.217.168.67) 56(84) bytes of data.
64 bytes from zrh04s15-in-f3.1e100.net (172.217.168.67): icmp_seq=1 ttl=54 time=10.0 ms
64 bytes from zrh04s15-in-f3.1e100.net (172.217.168.67): icmp_seq=2 ttl=54 time=10.3 ms
64 bytes from zrh04s15-in-f3.1e100.net (172.217.168.67): icmp_seq=3 ttl=54 time=9.73 ms
64 bytes from zrh04s15-in-f3.1e100.net (172.217.168.67): icmp_seq=4 ttl=54 time=9.90 ms
64 bytes from zrh04s15-in-f3.1e100.net (172.217.168.67): icmp_seq=5 ttl=54 time=9.69 ms
64 bytes from zrh04s15-in-f3.1e100.net (172.217.168.67): icmp_seq=6 ttl=54 time=9.94 ms
^C
--- www.google.ch ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 9.691/9.947/10.349/0.225 ms
root@Client_in_LAN:/#
```

f. Remarques (sur le message du premier ping)?

Réponse

Nous avons **ping** www.google.ch et grâce à la résolution DNS Client_In_Lan a résolu www.google.ch en 172.217.168.67.

Règles pour les protocoles HTTP et HTTPS

Créer et appliquer les règles adéquates pour que les **conditions 3 et 4 du cahier des charges** soient respectées. Tester que les règles soient fonctionnelles en utilisant wget depuis le Client_in_LAN pour télécharger une ressource depuis un site Web de votre choix (sur le WAN). Par exemple :

```
wget http://www.heig-vd.ch
```

- Créer et appliquer les règles adéquates avec des commandes iptables.

Commandes iptables :

```
#-----
# 3. HTTP
#-----
echo "3. Allows HTTP Connection for LAN"

# LAN --> WAN
$docker iptables -A FORWARD -s 192.168.100.0/24 -i eth2 -p tcp --dport 80 -
-j ACCEPT
$docker iptables -A FORWARD -s 192.168.100.0/24 -i eth2 -p tcp --dport 8080
-j ACCEPT

#-----
# 4. HTTPS
#-----
echo "4. Allows HTTPS secure Connection for LAN"
```

```
$docker iptables -A FORWARD -s 192.168.100.0/24 -i eth2 -p tcp --dport 443 -j ACCEPT
```

- Créer et appliquer les règles adéquates avec des commandes iptables pour que la **condition 5 du cahier des charges** soit respectée.

Commandes iptables :

```
#-----
# 5. HTTP DMZ
#-----
echo "5. Allows to reach DMZ on port 80 from LAN and WAN"

# LAN --> DMZ.3
$docker iptables -A FORWARD -d 192.168.200.3 -i eth2 -p tcp --sport 80 -j ACCEPT
# WAN --> DMZ.3
$docker iptables -A FORWARD -d 192.168.200.3 -i eth0 -p tcp --sport 80 -j ACCEPT
```

- g. Tester l'accès à ce serveur depuis le LAN utilisant utilisant wget (ne pas oublier les captures d'écran).

```
root@Client in LAN:/# wget 192.168.200.3
--2019-03-26 20:37:43-- http://192.168.200.3/
Connecting to 192.168.200.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 612 [text/html]
Saving to: 'index.html.1'

index.html.1          100%[=====>]
 612  --.-KB/s    in 0s

2019-03-26 20:37:43 (57.8 MB/s) - 'index.html.1' saved [612/612]

root@Client in LAN:/#
```

Règles pour le protocole ssh

- h. Créer et appliquer la règle adéquate pour que les ****conditions 6 et 7 du cahier des charges**** soient respectées.

Commandes iptables :

```
#-----  
# 6. SSH DMZ  
#-----  
echo "6. Allows to admin DMZ with SSH from LAN"  
  
# LAN --> DMZ.3  
$docker iptables -A FORWARD -s 192.168.100.3 -d 192.168.200.3 -i eth2 -p  
tcp --dport 22 -j ACCEPT  
  
#-----  
# 7. SSH FIREWALL  
#-----  
echo "7. Allows to admin FIREWALL with SSH from LAN"  
  
# LAN --> FIREWALL  
$docker iptables -A INPUT -s 192.168.100.3 -i eth2 -p tcp --dport 22 -j  
ACCEPT
```

Depuis le client dans le LAN, tester l'accès avec la commande suivante :

```
ssh root@192.168.200.3 (password : celui que vous avez configuré)
```

```
root@Client_in_LAN:/# ssh root@192.168.200.3  
root@192.168.200.3's password:  
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-46-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
This system has been minimized by removing packages and content that are  
not required on a system that users do not log into.  
  
To restore this content, you can run the 'unminimize' command.  
Last login: Tue Mar 26 20:43:20 2019 from 192.168.100.3  
root@Server_in_DMZ:~#
```

i. Expliquer l'utilité de **ssh** sur un serveur.

Réponse

Grâce à un accès SSH nous sommes capables de configurer un serveur à distance. Cela est très pratique quand ce dernier est dépourvu d'écran ou bien inatteignable physiquement.

- j. En général, à quoi faut-il particulièrement faire attention lors de l'écriture des règles du pare-feu pour ce type de connexion ?

Réponse

Il faut faire attention à configurer le bon port sur la bonne interface. Dans notre cas une petite erreur pourrait exposer notre firewall à internet. En effet si nous avons configuré l'interface eth0 (172.17.0.2) et non eth2 (192.168.100.2) des pirates mal intentionnés auraient pu attaquer notre firewall!

Règles finales iptables

A présent, vous devriez avoir le matériel nécessaire afin de reproduire la table de filtrage que vous avez conçue au début de ce laboratoire.

- j. Insérer la capture d'écran avec toutes vos règles iptables
-

```

root@Firewall:/# iptables -L
Chain INPUT (policy DROP)
target      prot opt source                destination          tcp dpt:22
ACCEPT      tcp  --  192.168.100.3          anywhere

Chain FORWARD (policy DROP)
target      prot opt source                destination          ctstate RELATED,ESTABLISHED
DROP        all  --  anywhere              anywhere             ctstate INVALID
ACCEPT      icmp --  192.168.100.0/24       anywhere             icmp echo-request
ACCEPT      icmp --  anywhere              192.168.100.0/24    icmp echo-reply
ACCEPT      icmp --  192.168.200.0/24       192.168.100.0/24    icmp echo-request
ACCEPT      icmp --  192.168.100.0/24       192.168.200.0/24    icmp echo-reply
ACCEPT      icmp --  192.168.200.0/24       192.168.100.0/24    icmp echo-request
ACCEPT      icmp --  192.168.100.0/24       192.168.200.0/24    icmp echo-reply
ACCEPT      udp  --  192.168.100.0/24       anywhere             udp dpt:53
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:53
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:80
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:8080
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:443
ACCEPT      tcp  --  anywhere              192.168.200.3       tcp spt:80
ACCEPT      tcp  --  anywhere              192.168.200.3       tcp spt:80
ACCEPT      tcp  --  192.168.100.3          192.168.200.3       tcp dpt:22

Chain OUTPUT (policy DROP)
target      prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT      all  --  anywhere              anywhere

root@Firewall:/#
root@Firewall:/# iptables -L
Chain INPUT (policy DROP)
target      prot opt source                destination          tcp dpt:22
ACCEPT      tcp  --  192.168.100.3          anywhere

Chain FORWARD (policy DROP)
target      prot opt source                destination          ctstate RELATED,ESTABLISHED
DROP        all  --  anywhere              anywhere             ctstate INVALID
ACCEPT      icmp --  192.168.100.0/24       anywhere             icmp echo-request
ACCEPT      icmp --  anywhere              192.168.100.0/24    icmp echo-reply
ACCEPT      icmp --  192.168.200.0/24       192.168.100.0/24    icmp echo-request
ACCEPT      icmp --  192.168.100.0/24       192.168.200.0/24    icmp echo-reply
ACCEPT      icmp --  192.168.200.0/24       192.168.100.0/24    icmp echo-request
ACCEPT      icmp --  192.168.100.0/24       192.168.200.0/24    icmp echo-reply
ACCEPT      udp  --  192.168.100.0/24       anywhere             udp dpt:53
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:53
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:80
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:8080
ACCEPT      tcp  --  192.168.100.0/24       anywhere             tcp dpt:443
ACCEPT      tcp  --  anywhere              192.168.200.3       tcp spt:80
ACCEPT      tcp  --  anywhere              192.168.200.3       tcp spt:80
ACCEPT      tcp  --  192.168.100.3          192.168.200.3       tcp dpt:22

Chain OUTPUT (policy DROP)
target      prot opt source                destination          ctstate RELATED,ESTABLISHED
ACCEPT      all  --  anywhere              anywhere

root@Firewall:/#

```