Teaching-HEIGVD-SRX-2021-Laboratoire-Firewall

Auteurs : Ryan Sauge, Dylan Canton

Travail à réaliser en équipes de deux personnes.

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 les ficher `Dockerfile` et `docker-compose.yml` indispensables 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

L'objectif principal de ce laboratoire est de familiariser les étudiants avec les pares-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, 2021, version 7.0. 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.

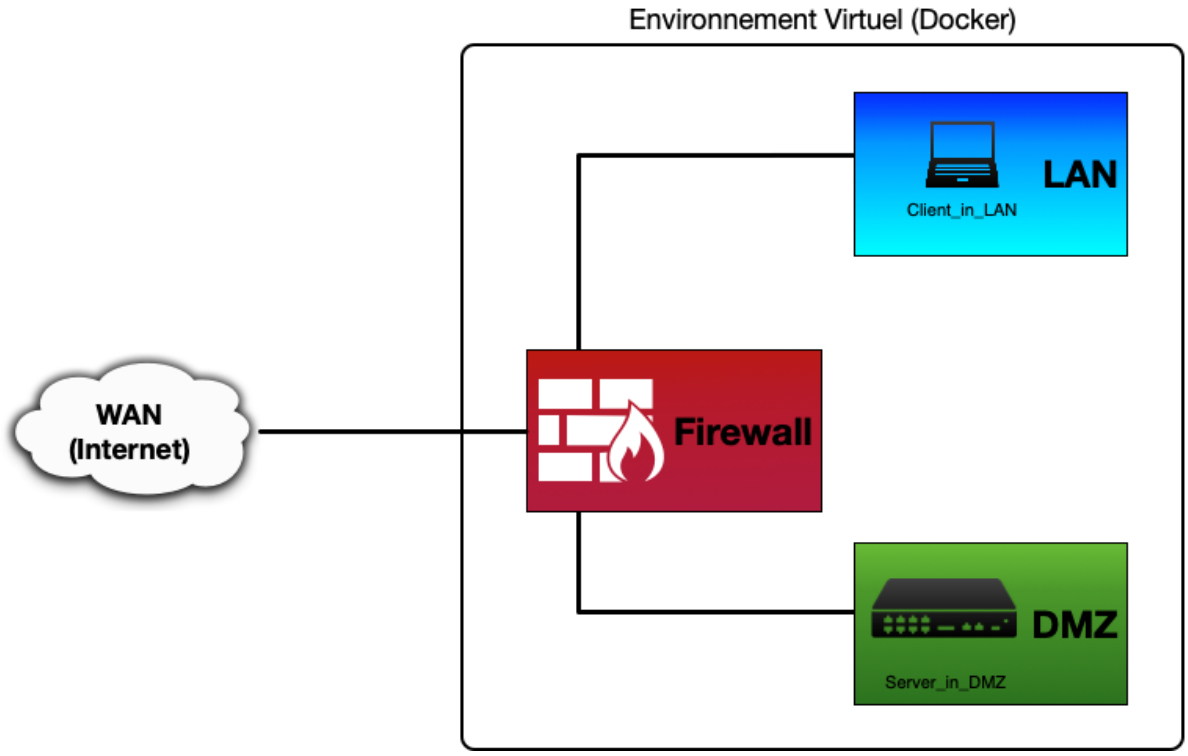
Echéance

Ce travail devra être rendu le dimanche après la fin de la 2ème séance de laboratoire, soit au plus tard, **le 01 avril 2021, à 23h59.**

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 ou votre propre réseau 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).

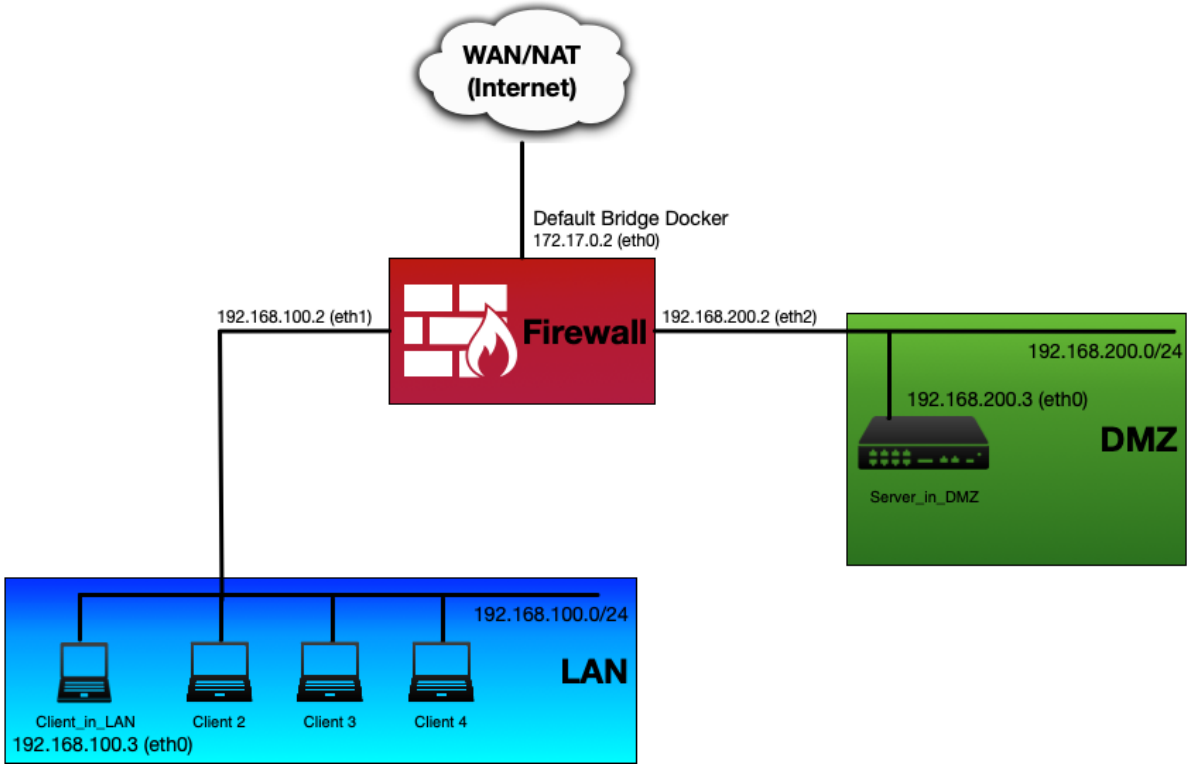
Nous allons utiliser les trois interfaces réseaux de la machine Firewall afin de pouvoir connecter le LAN et la DMZ à Internet (WAN). Les machines Client_In_LAN et Server_In_DMZ comportent chacune une interfaces réseau eth0.

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 ou à votre propre 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
192.168.100.0/24 (LAN)	interface WAN	UDP	*	53	ACCEPT
interface WAN	192.168.100.0/24 (LAN)	UDP	53	*	ACCEPT
192.168.100.0/24 (LAN)	interface WAN	TCP	*	53	ACCEPT
interface WAN	192.168.100.0/24 (LAN)	TCP	53	*	ACCEPT
192.168.100.0/24 (LAN)	interface WAN	ICMP	*	*	ACCEPT
interface WAN	192.168.100.0/24 (LAN)	ICMP	*	*	ACCEPT
192.168.100.0/24 (LAN)	192.168.200.0/24 (DMZ)	ICMP	*	*	ACCEPT
192.168.200.0/24 (DMZ)	192.168.100.0/24 (LAN)	ICMP	*	*	ACCEPT
192.168.200.0/24 (DMZ)	192.168.100.0/24 (LAN)	ICMP	*	*	ACCEPT
192.168.100.0/24 (LAN)	192.168.200.0/24 (DMZ)	ICMP	*	*	ACCEPT
192.168.100.0/24 (LAN)	interface WAN	TCP	*	80	ACCEPT
interface WAN	192.168.100.0/24 (LAN)	TCP	80	*	ACCEPT
192.168.100.0/24 (LAN)	interface WAN	TCP	*	8080	ACCEPT
interface WAN	192.168.100.0/24 (LAN)	TCP	8080	*	ACCEPT
192.168.100.0/24 (LAN)	interface WAN	TCP	*	443	ACCEPT
interface WAN	192.168.100.0/24 (LAN)	TCP	443	*	ACCEPT
192.168.100.0/24 (LAN)	192.168.200.3 (SERVEUR WEB)	TCP	*	80	ACCEPT
192.168.200.3 (SERVEUR WEB)	192.168.100.0/24 (LAN)	TCP	80	*	ACCEPT
interface WAN	192.168.200.3 (SERVEUR WEB)	TCP	*	80	ACCEPT
192.168.200.3 (SERVEUR WEB)	interface WAN	TCP	80	*	ACCEPT
192.168.100.3 (CLIENT LAN)	192.168.200.3 (SERVEUR WEB)	TCP	*	22	ACCEPT

Adresse IP source	Adresse IP destination	Type	Port src	Port dst	Action
192.168.200.3 (SERVEUR WEB)	192.168.100.3 (CLIENT LAN)	TCP	22	*	ACCEPT
192.168.100.3 (CLIENT LAN)	FIREWALL	TCP	*.	22	ACCEPT
192.168.100.2 (FIREWALL)	FIREWALL	TCP	22	*	ACCEPT
*	*	*	*	*	DROP

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 a été testée avec la version 3.2.2 de Docker Desktop pour Mac. Si vous rencontrez des problèmes, une mise à jour de Docker es peut-être la solution.

Vous pouvez trouver Docker pour Windows et Mac OS ici.

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 ici.

Démarrage de l'environnement virtuel

Ce laboratoire utilise docker-compose, un outil pour la gestion d'applications utilisant multiples conteneurs. Il va se charger de créer les réseaux `lan` et `dmz`, la machine Firewall, un serveur dans le réseau DMZ et une machine dans le réseau LAN et de tout interconnecter correctement.

Nous allons commencer par lancer docker-compose. Il suffit de taper la commande suivante dans le répertoire racine du labo (celui qui contient le fichier `docker-compose.yml`:

```
docker-compose up --detach
```

Le téléchargement et génération d'images prend peu de temps.

Vous pouvez vérifier que les réseaux ont été créés avec la commande `docker network ls`. Un réseau `lan` et un réseau `dmz` 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 de base. `docker-compose` l'utilise comme un modèle pour générer les conteneurs. Vous pouvez vérifier que les trois conteneurs sont crées et qu'ils fonctionnent à l'aide de la commande suivante.

```
docker ps
```

Communication avec les conteneurs et configuration du firewall

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

- Firewall
- Client_in_LAN
- Server_in_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 !

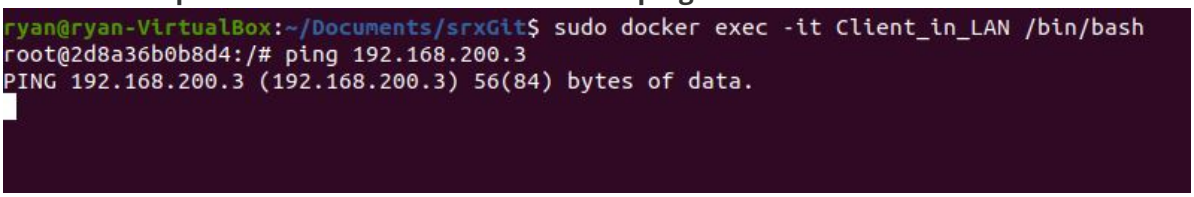
Configuration de base

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.



En effet, la communication entre les clients dans le LAN et les serveurs dans la DMZ doit passer à travers le Firewall. Dans certaines configuration, il est probable que le ping arrive à passer par le bridge par défaut. Ceci est une limitation de Docker. **Si votre ping passe**, vous pouvez accompagner votre capture du ping avec une capture d'une commande traceroute qui montre que le ping ne passe pas actuellement par le Firewall mais qu'il a emprunté un autre chemin.

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 dans DMZ, taper les commandes suivantes :

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

service nginx start
service ssh start
```

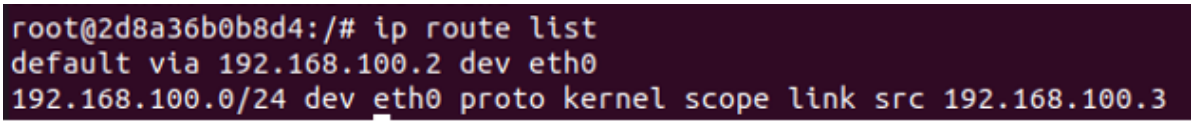
Les deux dernières commandes démarrent les services Web et SSH du serveur.

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

```
ping 192.168.100.3
```

LIVRABLES : captures d'écran des routes des deux machines et de votre nouvelle tentative de ping.

Route client :



Route serveur

```
root@79832655c50d:/# ip route list
default via 192.168.200.2 dev eth0
192.168.200.0/24 dev eth0 proto kernel scope link src 192.168.200.3
```

Ping effectué

```
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=63 time=0.090 ms
64 bytes from 192.168.100.3: icmp_seq=2 ttl=63 time=0.102 ms
64 bytes from 192.168.100.3: icmp_seq=3 ttl=63 time=0.058 ms
64 bytes from 192.168.100.3: icmp_seq=4 ttl=63 time=0.078 ms
64 bytes from 192.168.100.3: icmp_seq=5 ttl=63 time=0.060 ms
64 bytes from 192.168.100.3: icmp_seq=6 ttl=63 time=0.104 ms
64 bytes from 192.168.100.3: icmp_seq=7 ttl=63 time=0.096 ms
64 bytes from 192.168.100.3: icmp_seq=8 ttl=63 time=0.061 ms
64 bytes from 192.168.100.3: icmp_seq=9 ttl=63 time=0.061 ms
64 bytes from 192.168.100.3: icmp_seq=10 ttl=63 time=0.060 ms
64 bytes from 192.168.100.3: icmp_seq=11 ttl=63 time=0.107 ms
64 bytes from 192.168.100.3: icmp_seq=12 ttl=63 time=0.060 ms
64 bytes from 192.168.100.3: icmp_seq=13 ttl=63 time=0.108 ms
64 bytes from 192.168.100.3: icmp_seq=14 ttl=63 time=0.065 ms
64 bytes from 192.168.100.3: icmp_seq=15 ttl=63 time=0.101 ms
^C
--- 192.168.100.3 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14331ms
rtt min/avg/max/mdev = 0.058/0.080/0.108/0.020 ms
```

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 ou sans un service de redirection ICMP. Vous pouvez le vérifier avec un ping depuis le client ou le serveur vers une adresse Internet.

Par exemple :

```
ping 8.8.8.8
```

Si votre ping passe mais que la réponse contient un *Redirect Host*, ceci indique que votre ping est passé grace à la redirection ICMP, mais que vous n'arrivez pas encore à contacter l'Internet à travers de Firewall. Ceci est donc aussi valable pour l'instant et accepté comme résultat.

LIVRABLE : capture d'écran de votre ping vers l'Internet. Un ping qui ne passe pas ou des réponses containant des *Redirect Host* sont acceptés.

```
root@79832655c50d:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 192.168.200.2: icmp_seq=2 Redirect Host(New nexthop: 192.168.200.1)
From 192.168.200.2: icmp_seq=3 Redirect Host(New nexthop: 192.168.200.1)
From 192.168.200.2: icmp_seq=4 Redirect Host(New nexthop: 192.168.200.1)
From 192.168.200.2: icmp_seq=5 Redirect Host(New nexthop: 192.168.200.1)
From 192.168.200.2: icmp_seq=6 Redirect Host(New nexthop: 192.168.200.1)
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6042ms
```

Configuration réseau du firewall

On va fournir une route vers l'internet à travers le firewall aux deux réseaux connectés :

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

Cette 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.

L'autre commande démarre le service SSH du serveur.

Vérifiez que la connexion à l'Internet est maintenant possible depuis les deux autres machines. Pas besoin de capture d'écran mais assurez vous que les pings passent sans besoin de redirection de host avant de continuer.

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 avez le choix d'utiliser le mode avec état, sans état ou une combinaison des deux**.

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
```

Récuperer 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

Tout d'abord, nous établissons les politiques générales qui sont de tout bloquer par défaut (**condition 8 du cahier des charges**) :

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

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 WAN,
- les ping depuis la DMZ vers le LAN.

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

Commandes iptables :

```
#LAN vers DMZ et réponse
iptables -A FORWARD -s  192.168.100.0/24 -d 192.168.200.0/24 -p icmp --icmp-type
8 -j ACCEPT
iptables -A FORWARD -s  192.168.200.0/24 -d 192.168.100.0/24 -p icmp --icmp-type
0 -j ACCEPT

#LAN vers WAN et réponse
iptables -A FORWARD -s  192.168.100.0/24 -o eth0 -p icmp --icmp-type 8 -j ACCEPT
iptables -A FORWARD -i eth0 -d 192.168.100.0/24 -p icmp --icmp-type 0 -j ACCEPT

#DMZ vers LAN et réponse
iptables -A FORWARD -s  192.168.200.0/24 -d 192.168.100.0/24 -p icmp --icmp-type
8 -j ACCEPT
iptables -A FORWARD -s  192.168.100.0/24 -d 192.168.200.0/24 -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
```

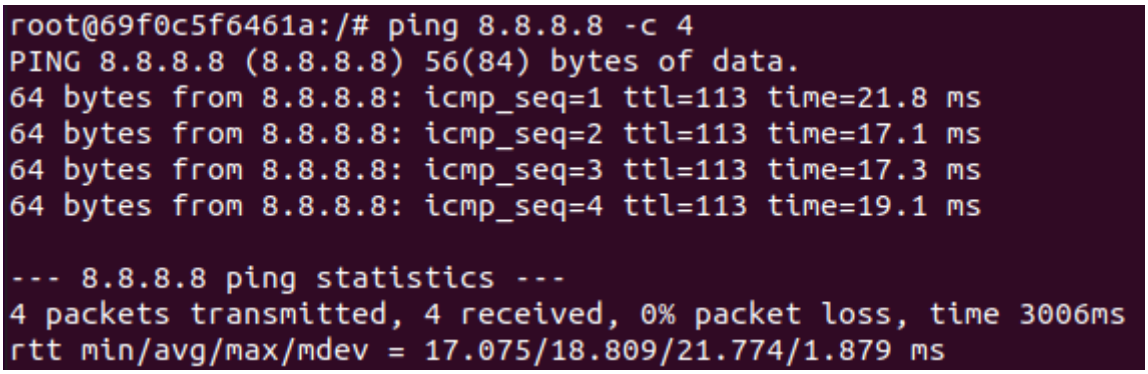
Faire une capture du ping.

Vérifiez aussi la route entre votre client et le service `8.8.8.8`. Elle devrait partir de votre client et traverser votre Firewall :

```
traceroute 8.8.8.8
```

LIVRABLE : capture d'écran du traceroute et de votre ping vers l'Internet. Il ne devrait pas y avoir des *Redirect Host* dans les réponses au ping !

Ping du LAN vers le WAN :

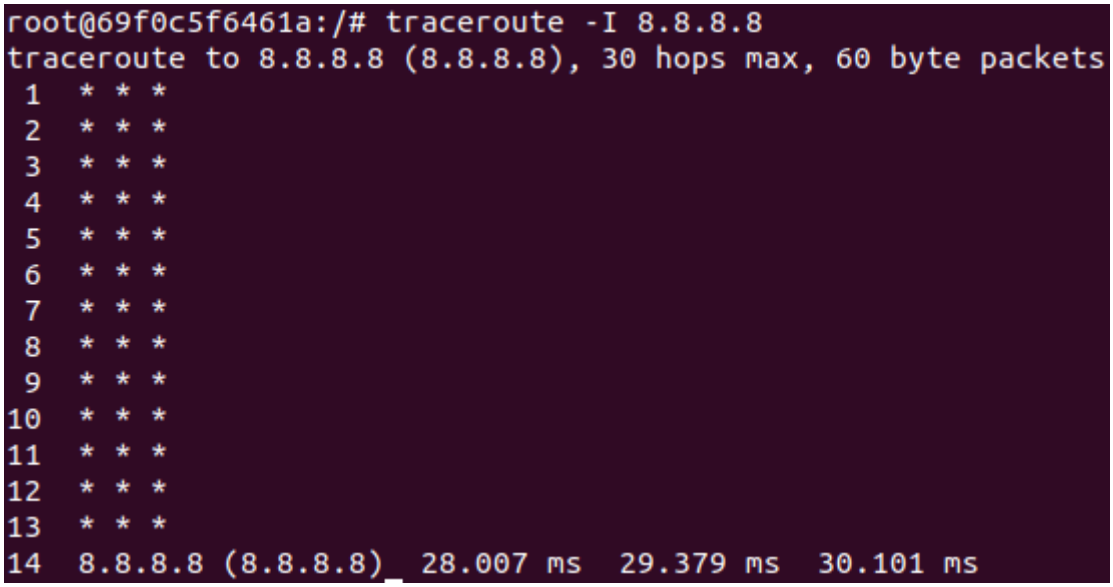


Traceroute du LAN vers le WAN :

Bien que le ping fonctionne, le traceroute n'affiche aucun nœud. Cela peut être dû à docker.

Cependant, en spécifiant le protocole ICMP dans la commande, traceroute aboutis au serveur 8.8.8.8

```
traceroute -I 8.8.8.8
```



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	Ajout d'une règle pour l'IP de l'interface si on veut que ca marche
Interface LAN du FW	KO	Ajout d'une règle pour l'IP de l'interface si on veut que ca marche
Client LAN	OK	-
Serveur WAN	OK	-

De Server_in_DMZ à	**OK/KO**	**Commentaires et explications**
Interface DMZ du FW	KO	Ajout d'une règle pour l'IP de l'interface si on veut que ca marche
Interface LAN du FW	KO	Ajout d'une règle pour l'IP de l'interface si on veut que ca marche
Serveur DMZ	OK	-
Serveur WAN	KO	Règle déjà ajoutée pour interdire le ping de la DMZ vers le WAN

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.

LIVRABLE : capture d'écran de votre ping.

```
root@69f0c5f6461a:/# ping www.google.com
ping: www.google.com: Temporary failure in name resolution
```

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

Commandes iptables :

```
#LAN vers WAN (UDP) et réponse
iptables -A FORWARD -p udp -s 192.168.100.0/24 -o eth0 --dport 53 -j ACCEPT
iptables -A FORWARD -p udp -i eth0  --sport 53 -d 192.168.100.0/24  -j ACCEPT

#LAN vers WAN (TCP) et réponse
iptables -A FORWARD -p tcp -s 192.168.100.0/24 -o eth0 --dport 53 -j ACCEPT
iptables -A FORWARD -p tcp -i eth0  --sport 53 -d 192.168.100.0/24  -j ACCEPT
```

---

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

LIVRABLE : capture d'écran de votre ping.

```
root@69f0c5f6461a:/# ping www.google.com -c 4
PING www.google.com (172.217.168.36) 56(84) bytes of data.
64 bytes from zrh04s14-in-f4.1e100.net (172.217.168.36): icmp_seq=1 ttl=113 time=19.7 ms
64 bytes from zrh04s14-in-f4.1e100.net (172.217.168.36): icmp_seq=2 ttl=113 time=25.4 ms
64 bytes from zrh04s14-in-f4.1e100.net (172.217.168.36): icmp_seq=3 ttl=113 time=19.7 ms
64 bytes from zrh04s14-in-f4.1e100.net (172.217.168.36): icmp_seq=4 ttl=113 time=22.2 ms

--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 19.652/21.754/25.413/2.346 ms
```

---

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

Réponse

LIVRABLE : Le service DNS utilise le port 53, ce port n'étant pas autorisé avant d'avoir établis la règle, il était alors impossible pour le client de résoudre le nom de domaine.

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 :

Nous avons configuré ces règles avec état car on estime important d'un point de vue sécuritaire qu'un serveur web ne puisse pas initier de connexion avec le client.

```
#LAN vers WAN (HTTP) et réponse
iptables -A FORWARD -p tcp -s 192.168.100.0/24 -o eth0 --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

iptables -A FORWARD -p tcp -i eth0 --sport 80 -d 192.168.100.0/24 -m conntrack --ctstate ESTABLISHED -j ACCEPT

#LAN vers WAN (HTTP) et réponse
iptables -A FORWARD -p tcp -s 192.168.100.0/24 -o eth0 --dport 8080 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

iptables -A FORWARD -p tcp -i eth0 --sport 8080 -d 192.168.100.0/24 -m conntrack --ctstate ESTABLISHED -j ACCEPT

#LAN vers WAN (HTTPS) et réponse
iptables -A FORWARD -p tcp -s 192.168.100.0/24 -o eth0 --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

iptables -A FORWARD -p tcp -i eth0 --sport 443 -d 192.168.100.0/24 -m conntrack --ctstate ESTABLISHED -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 :

```
#LAN vers Serveur Web (DMZ) et réponse
iptables -A FORWARD -p tcp -s 192.168.100.0/24 -d 192.168.200.3 --dport 80  -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.200.3 --sport 80 -d 192.168.100.0/24  -j ACCEPT

#WAN vers Serveur Web (DMZ) et réponse
iptables -A FORWARD -p tcp -i eth0 -d 192.168.200.3 --dport 80  -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.200.3  --sport 80 -o eth0 -j ACCEPT
```

---

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

LIVRABLE : capture d'écran.

```
root@69f0c5f6461a:/# wgets 192.168.200.3
bash: wgets: command not found
root@69f0c5f6461a:/# wget 192.168.200.3
--2021-03-25 15:27:04--  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.2'

index.html.2                    100%[=======================================================>]     612  --.-KB/s    in 0s

2021-03-25 15:27:04 (65.2 MB/s) - 'index.html.2' saved [612/612]
```

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 :

Nous avons configuré ces règles avec état car on estime important d'un point de vue sécuritaire que seul le client ssh puisse initier la connexion. On évite ainsi que le serveur web, présente dans la DMZ ou le Firewall ne puissent initier la connexion ssh avec le client(et donc le contrôler à distance).

```
#Client LAN vers Serveur Web (DMZ) et réponse
iptables -A FORWARD -p tcp -s 192.168.100.3 -d 192.168.200.3 --dport 22 -m
conntrack  --ctstate NEW,ESTABLISHED -j ACCEPT

iptables -A FORWARD -p tcp -s 192.168.200.3 --sport 22 -d 192.168.100.3 -m
conntrack  --ctstate ESTABLISHED -j ACCEPT

#Client LAN vers Firewall et réponse
iptables -A INPUT -p tcp -s 192.168.100.3 --dport 22 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -p tcp --sport 22 -d 192.168.100.3 -m conntrack --ctstate
ESTABLISHED -j ACCEPT
```

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

```
ssh root@192.168.200.3
```

LIVRABLE : capture d'écran de votre connexion ssh.

Connexion SSH du lient LAN vers le serveur Web (DMZ) :

```
root@69f0c5f6461a:/# ssh root@192.168.200.3
Last login: Thu Mar 25 17:27:43 2021 from 192.168.200.2
root@bc2cbda14a7d:~# exit
logout
Connection to 192.168.200.3 closed.
root@69f0c5f6461a:/#
```

Connexion SSH du client LAN vers le Firewall :

```
root@69f0c5f6461a:/# ssh root@192.168.100.2
Last login: Thu Mar 25 17:40:45 2021 from 192.168.100.3
root@c9f851667a34:~# exit
logout
Connection to 192.168.100.2 closed.
root@69f0c5f6461a:/#
```

---

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

Réponse

LIVRABLE : SSH permet d'établir une connexion entre le client et le serveur de manière sécurisée. L'avantage est donc de pouvoir configurer et administrer le serveur à distance.

Les données sont chiffrées et le protocole est mieux sécurisée que par exemple avec telnet. On évite aussi dans ce cas la des attaques de type MITM car le serveur s'authentifie auprès du client et le client s'authentifie auprès du serveur.

---

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

LIVRABLE : Il faut écrire les règles de telle manière que seul le client concerné puisse établir une connexion. Sinon, il y a le risque que des pirates tentent une attaque brute-force sur le serveur pour trouver une paire utilisateur/mot de passe valide.

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

---

LIVRABLE : capture d'écran avec toutes vos règles.

```
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.100.3        anywhere             tcp dpt:ssh ctstate NEW,ESTABLISHED

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  192.168.100.0/24     192.168.200.0/24     icmp echo-request
ACCEPT     icmp --  192.168.200.0/24     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 --  192.168.100.0/24     anywhere             icmp echo-request
ACCEPT     icmp --  anywhere             192.168.100.0/24     icmp echo-reply
ACCEPT     udp  --  192.168.100.0/24     anywhere             udp dpt:domain
ACCEPT     udp  --  anywhere             192.168.100.0/24     udp spt:domain
ACCEPT     tcp  --  192.168.100.0/24     anywhere             tcp dpt:domain
ACCEPT     tcp  --  anywhere             192.168.100.0/24     tcp spt:domain
ACCEPT     tcp  --  192.168.100.0/24     anywhere             tcp dpt:http ctstate NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.100.0/24     tcp spt:http ctstate ESTABLISHED
ACCEPT     tcp  --  192.168.100.0/24     anywhere             tcp dpt:http-alt ctstate NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.100.0/24     tcp spt:http-alt ctstate ESTABLISHED
ACCEPT     tcp  --  192.168.100.0/24     anywhere             tcp dpt:https ctstate NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             192.168.100.0/24     tcp spt:https ctstate ESTABLISHED
ACCEPT     tcp  --  192.168.100.0/24     192.168.200.3        tcp dpt:http
ACCEPT     tcp  --  192.168.200.3        192.168.100.0/24     tcp spt:http
ACCEPT     tcp  --  anywhere             192.168.200.3        tcp dpt:http
ACCEPT     tcp  --  192.168.200.3        anywhere             tcp spt:http
ACCEPT     tcp  --  192.168.100.3        192.168.200.3        tcp dpt:ssh ctstate NEW,ESTABLISHED
ACCEPT     tcp  --  192.168.200.3        192.168.100.3        tcp spt:ssh ctstate ESTABLISHED

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             192.168.100.3        tcp spt:ssh ctstate ESTABLISHED
```