

Teaching-HEIGVD-SRX-2021-Laboratoire-Firewall

Travail à réaliser en équipes de deux personnes.

AUTEURS: Alexandra Cerottini & Fiona Gamboni

DATE: 21.03.2021

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 fichier `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](#)

[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, 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.

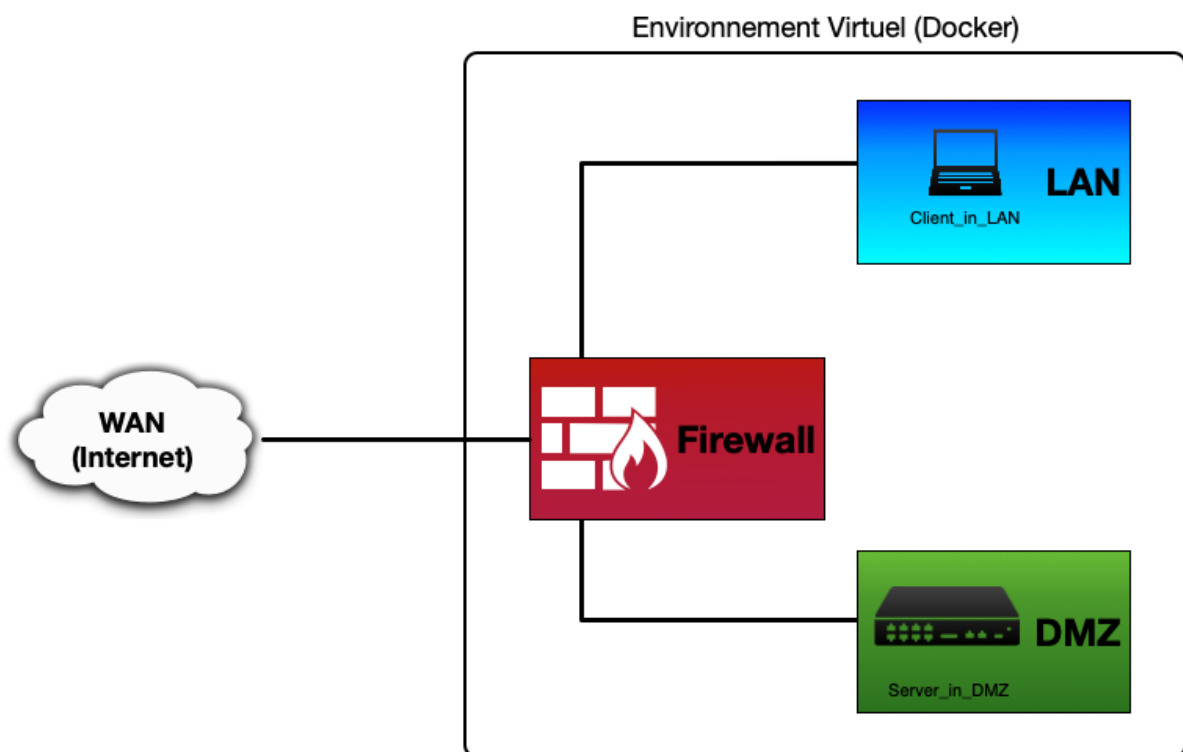
Echéance

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.

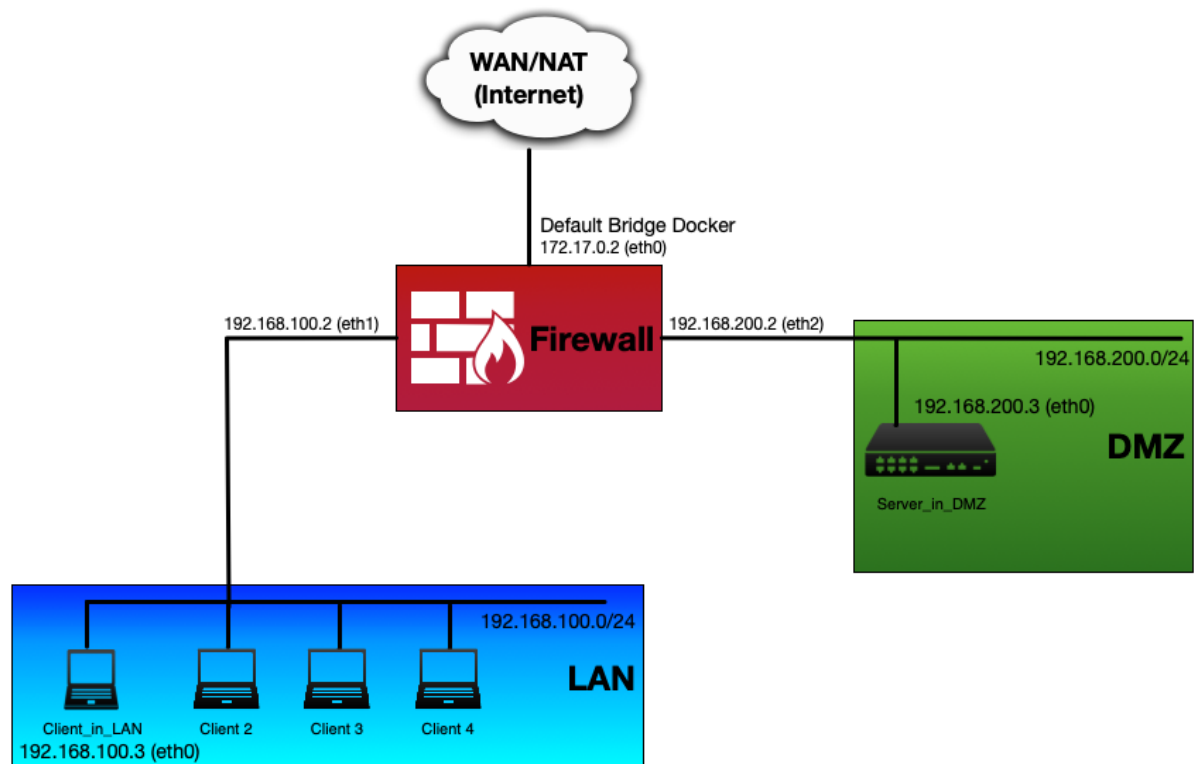
Plan d'adressage

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	WAN	TCP	*	53	Accept
WAN	192.168.100.0/24	TCP	53	*	Accept
192.168.100.0/24	WAN	UDP	*	53	Accept
WAN	192.168.100.0/24	UDP	53	*	Accept
192.168.100.0/24	WAN	ICMP-8			Accept
WAN	192.168.100.0/24	ICMP-0			Accept
192.168.100.0/24	192.168.200.0/24	ICMP-8			Accept
192.168.200.0/24	192.168.100.0/24	ICMP-0			Accept
192.168.200.0/24	192.168.100.0/24	ICMP-8			Accept
192.168.100.0/24	192.168.200.0/24	ICMP-0			Accept
192.168.100.0/24	WAN	TCP	*	80	Accept
192.168.100.0/24	WAN	TCP	*	8080	Accept
WAN	192.168.100.0/24	TCP	80	*	Accept
192.168.100.0/24	WAN	TCP	8080	*	Accept
192.168.100.0/24	WAN	TCP	*	443	Accept
WAN	192.168.100.0/24	TCP	443	*	Accept
192.168.100.0/24	192.168.200.3	TCP	*	80	Accept
192.168.200.3	192.168.100.0/24	TCP	80	*	Accept
WAN	192.168.200.3	TCP	*	80	Accept
192.168.200.3	WAN	TCP	80	*	Accept
192.168.100.3	192.168.200.3	TCP	*	22	Accept
192.168.200.3	192.168.100.3	TCP	22	*	Accept
192.168.100.3	192.168.100.2	TCP	*	22	Accept
192.168.100.2	192.168.100.3	TCP	22	*	Accept
*	192.168.100.0/24	any	*	*	Drop
*	192.168.200.0/24	any	*	*	Drop

Adresse IP source	Adresse IP destination	Type	Port src	Port dst	Action
192.168.100.0/24	*	any	*	*	Drop
192.168.200.0/24	*	any	*	*	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 créés 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 configurés 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.

```
root@33086d1de14b:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms

root@33086d1de14b:/#
```

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.


```
root@e39f7455d58c:/# 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=63 time=0.062 ms
64 bytes from 192.168.100.3: icmp_seq=2 ttl=63 time=0.060 ms
64 bytes from 192.168.100.3: icmp_seq=3 ttl=63 time=0.094 ms
64 bytes from 192.168.100.3: icmp_seq=4 ttl=63 time=0.181 ms
64 bytes from 192.168.100.3: icmp_seq=5 ttl=63 time=0.059 ms
^C
--- 192.168.100.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4102ms
rtt min/avg/max/mdev = 0.059/0.091/0.181/0.046 ms
root@e39f7455d58c:/#
```

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 grâce 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 contenant des *Redirect Host* sont acceptés.

Depuis le serveur (émet une redirection):

```
root@e39f7455d58c:/# 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)
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4091ms

root@e39f7455d58c:/#
```

Depuis le client:

```
root@33086d1de14b:/# 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 ---
8 packets transmitted, 0 received, 100% packet loss, time 7149ms

root@33086d1de14b:/#
```

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
```

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

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

Commandes iptables :

LIVRABLE : Commandes iptables

```
iptables -P INPUT DROP
```

```
iptables -P OUTPUT DROP
```

```
iptables -P FORWARD DROP
```

```
iptables -A FORWARD -p icmp --icmp-type 8 -s 192.168.100.0/24 -d  
192.168.200.0/24 -j ACCEPT
```

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

```
iptables -A FORWARD -p icmp --icmp-type 8 -s 192.168.100.0/24 -o eth0 -j ACCEPT
```

```
iptables -A FORWARD -p icmp --icmp-type 0 -i eth0 -d 192.168.100.0/24 -j ACCEPT
```

```
iptables -A FORWARD -p icmp --icmp-type 8 -s 192.168.200.0/24 -d  
192.168.100.0/24 -j ACCEPT
```

```
iptables -A FORWARD -p icmp --icmp-type 0 -s 192.168.100.0/24 -d  
192.168.200.0/24 -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 :

```
tracert 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:

```
root@33086d1de14b:/# 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=114 time=27.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=25.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=19.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=20.1 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 19.384/23.166/27.906/3.563 ms
root@33086d1de14b:/#
```

tracert:

Pour que la commande traceroute fonctionne, il faut mettre la FORWARD chain et l'OUTPUT chain en ACCEPT.

```
root@e44ca9fa6f12:/# traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1 Firewall.lan (192.168.100.2)  0.469 ms  0.325 ms  0.307 ms
 2 192.168.200.1 (192.168.200.1)  0.294 ms  0.257 ms  0.238 ms
 3 compalhub.home (192.168.0.1)  4.108 ms  5.064 ms  6.095 ms
 4 CH-NAX01A-RT1.aorta.net (84.116.251.239)  31.351 ms  31.336 ms  33.462 ms
 5 ch-nax01a-rc1-ae-20-0.aorta.net (84.116.204.53)  38.021 ms  36.985 ms  36.939 ms
 6 ch-zrh03a-rc1-ae-0-0.aorta.net (84.116.130.2)  39.057 ms  ch-ott01b-rc1-ae-1-0.aorta.net (84.116.130.14)  19.483 ms  ch-zrh03a-rc1-ae-0-0.aorta.net (84.116.130.2)  23.651 ms
 7 ch-zrh01b-ral-ae-1-0.aorta.net (84.116.134.142)  23.632 ms  28.650 ms  ch-zrh01b-ral-ae-9-0.aorta.net (84.116.134.22)  28.618 ms
 8 72.14.221.112 (72.14.221.112)  28.602 ms  22.665 ms  27.335 ms
 9 74.125.243.145 (74.125.243.145)  26.116 ms  26.100 ms  74.125.243.129 (74.125.243.129)  31.406 ms
10 64.233.175.167 (64.233.175.167)  31.420 ms  172.253.50.3 (172.253.50.3)  32.385 ms  172.253.50.1 (172.253.50.1)  31.404 ms
11 dns.google (8.8.8.8)  37.345 ms  37.331 ms  20.148 ms
```

- 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	INPUT et OUTPUT du firewall sont en DROP
Interface LAN du FW	KO	INPUT et OUTPUT du firewall sont en DROP
Client LAN	OK	On est sûr le même réseau
Serveur WAN	OK	Nous avons accepté les pings sur le WAN depuis le client

De Server_in_DMZ à	OK/KO	Commentaires et explications
Interface DMZ du FW	KO	INPUT et OUTPUT du firewall sont en DROP
Interface LAN du FW	KO	INPUT et OUTPUT du firewall sont en DROP
Serveur DMZ	OK	On est sûr le même réseau
Serveur WAN	KO	Nous n'avons pas accepté les pings sur le WAN depuis le serveur

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@33086d1de14b:/# ping www.google.com
ping: www.google.com: Temporary failure in name resolution
root@33086d1de14b:/#
```

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

Commandes iptables :

LIVRABLE : Commandes iptables

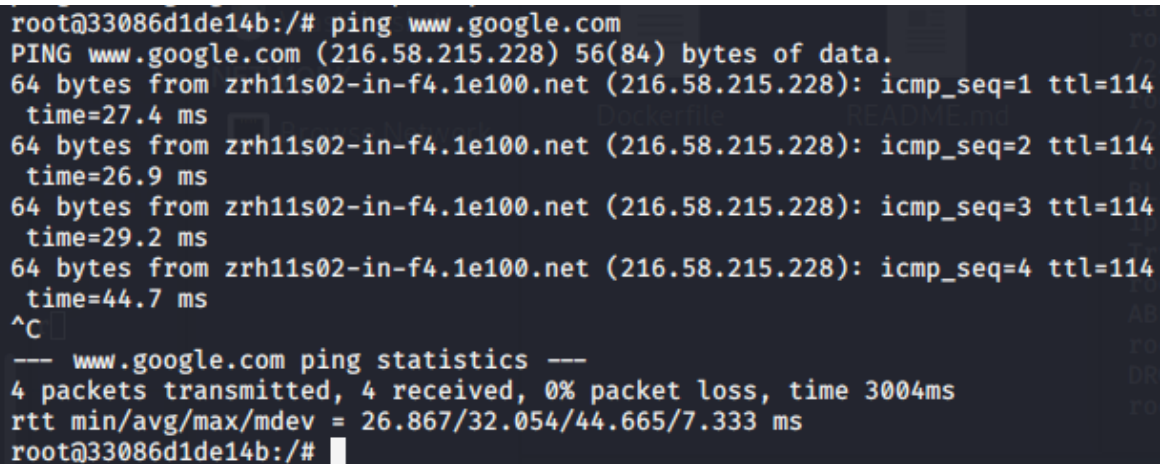
```
iptables -A FORWARD -p tcp --dport 53 -s 192.168.100.0/24 -o eth0 -j ACCEPT
iptables -A FORWARD -p udp --dport 53 -s 192.168.100.0/24 -o eth0 -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -i eth0 -d 192.168.100.0/24 -j ACCEPT
```

```
iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate INVALID -j DROP
```

(à partir d'ici, les retours du FORWARD seront gérés par le module conntrack)

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

LIVRABLE : capture d'écran de votre ping.



```
root@33086d1de14b:/# ping www.google.com
PING www.google.com (216.58.215.228) 56(84) bytes of data.
64 bytes from zrh11s02-in-f4.1e100.net (216.58.215.228): icmp_seq=1 ttl=114
time=27.4 ms
64 bytes from zrh11s02-in-f4.1e100.net (216.58.215.228): icmp_seq=2 ttl=114
time=26.9 ms
64 bytes from zrh11s02-in-f4.1e100.net (216.58.215.228): icmp_seq=3 ttl=114
time=29.2 ms
64 bytes from zrh11s02-in-f4.1e100.net (216.58.215.228): icmp_seq=4 ttl=114
time=44.7 ms
^C
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 26.867/32.054/44.665/7.333 ms
root@33086d1de14b:/#
```

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

Réponse

LIVRABLE : Votre réponse ici...

Comme la résolution du nom de domaine a échoué, le nom du domaine n'a pas pu être traduit en adresse IP. Le ping ne peut donc pas être exécuté.

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 :

LIVRABLE : Commandes iptables

```
iptables -A FORWARD -p tcp --dport 80 -s 192.168.100.0/24 -o eth0 -j ACCEPT
iptables -A FORWARD -p tcp --dport 8080 -s 192.168.100.0/24 -o eth0 -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -s 192.168.100.0/24 -o eth0 -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 :

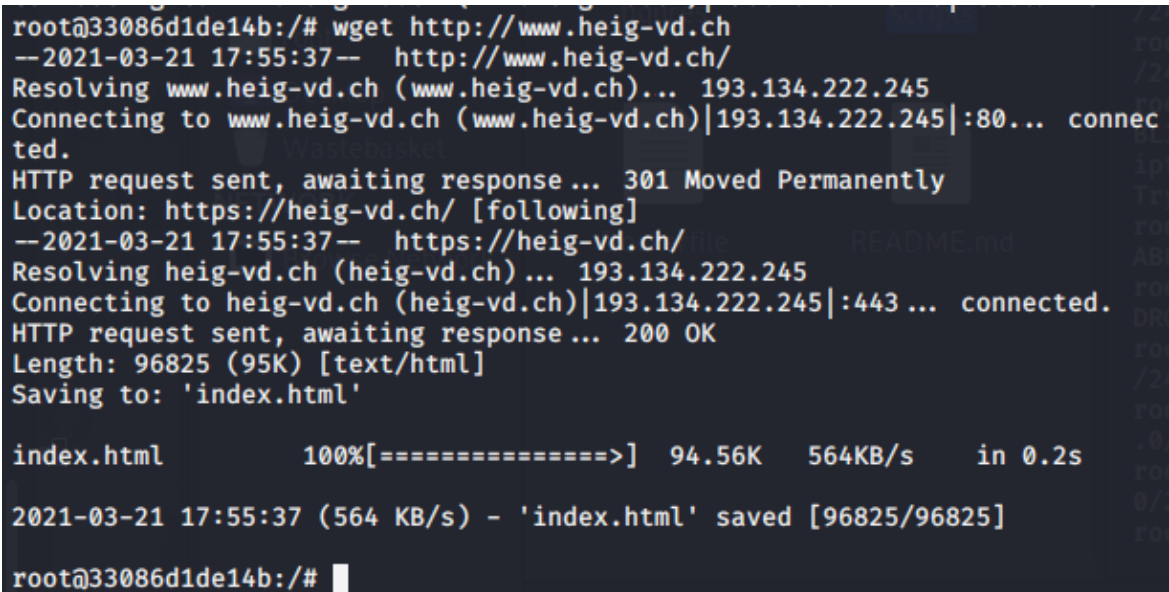
LIVRABLE : Commandes iptables

```
iptables -A FORWARD -p tcp --dport 80 -s 192.168.100.0/24 -d 192.168.200.3 -j
ACCEPT
iptables -A FORWARD -p tcp --dport 80 -i eth0 -d 192.168.200.3 -j ACCEPT
```

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

LIVRABLE : capture d'écran.

Pour les conditions 3 et 4:



```
root@33086d1de14b:/# wget http://www.heig-vd.ch
--2021-03-21 17:55:37-- http://www.heig-vd.ch/
Resolving www.heig-vd.ch (www.heig-vd.ch)... 193.134.222.245
Connecting to www.heig-vd.ch (www.heig-vd.ch)|193.134.222.245|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://heig-vd.ch/ [following]
--2021-03-21 17:55:37-- https://heig-vd.ch/
Resolving heig-vd.ch (heig-vd.ch)... 193.134.222.245
Connecting to heig-vd.ch (heig-vd.ch)|193.134.222.245|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 96825 (95K) [text/html]
Saving to: 'index.html'

index.html      100%[=====>]  94.56K  564KB/s   in 0.2s
2021-03-21 17:55:37 (564 KB/s) - 'index.html' saved [96825/96825]

root@33086d1de14b:/#
```

Pour la condition 5:


```
root@33086d1de14b:/# wget 192.168.200.3
--2021-03-21 18:13:39-- 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

2021-03-21 18:13:39 (127 MB/s) - 'index.html.1' saved [612/612]

root@33086d1de14b:/#
```

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 :

LIVRABLE : Commandes iptables

```
iptables -A FORWARD -p tcp --dport 22 -s 192.168.100.3 -d 192.168.200.3 -j
ACCEPT
```

```
iptables -A INPUT -p tcp --dport 22 -s 192.168.100.3 -d 192.168.100.2 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 22 -s 192.168.100.3 -d 192.168.100.2 -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.

Lorsqu'on se connecte au serveur de la DMZ:

```
root@e44ca9fa6f12:/# ssh root@192.168.200.3
The authenticity of host '192.168.200.3 (192.168.200.3)' can't be established.
ECDSA key fingerprint is SHA256:IVWvE5UcNKEc3SsfVjklLuGMJHxyHouaAliUdVuKjk4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.200.3' (ECDSA) to the list of known hosts.
```

```
root@e44ca9fa6f12:/# ssh root@192.168.200.3
Last login: Sun Mar 21 19:09:08 2021 from 192.168.200.2
root@ba5a8bfc7642:~#
```

Lorsqu'on se connecte au firewall:


```
root@33086d1de14b:/# ssh root@192.168.100.2
The authenticity of host '192.168.100.2 (192.168.100.2)' can't be established.
ECDSA key fingerprint is SHA256:LfAXhdhnp/ZISnY89WX5l0KWCukFFdpGHX1e+HMY+xY
.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.100.2' (ECDSA) to the list of known hosts.
root@e5f5069a6631:~#
```

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

Réponse

LIVRABLE : Votre réponse ici...

Le protocole SSH permet de se connecter de manière sécurisée sur des systèmes distants. Une fois connecté, tous les segments TCP sont authentifiés et chiffrés.

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 : Votre réponse ici...

Comme une connexion par ssh permet de prendre le contrôle total du serveur, il est important limiter le plus possible les moyens de connections. On autorise seulement le strict nécessaire au niveau des interfaces et ports.

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.

```

root@e5f5069a6631:/# iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source                destination            tcp dpt:22
ACCEPT     tcp  --  192.168.100.3          192.168.100.2

Chain FORWARD (policy DROP)
target     prot opt source                destination            icmp type 8
ACCEPT     icmp --  192.168.100.0/24       192.168.200.0/24       icmp type 0
ACCEPT     icmp --  192.168.200.0/24       192.168.100.0/24       icmp type 0
ACCEPT     icmp --  0.0.0.0/0              192.168.100.0/24       icmp type 0
ACCEPT     icmp --  192.168.100.0/24       0.0.0.0/0              icmp type 8
ACCEPT     icmp --  192.168.200.0/24       192.168.100.0/24       icmp type 8
ACCEPT     icmp --  192.168.100.0/24       192.168.200.0/24       icmp type 0
ACCEPT     tcp  --  192.168.100.0/24       0.0.0.0/0              tcp dpt:53
ACCEPT     udp  --  192.168.100.0/24       0.0.0.0/0              udp dpt:53
ACCEPT     all  --  0.0.0.0/0              0.0.0.0/0              ctstate RELATED,ESTABLISHED
DROP       all  --  0.0.0.0/0              0.0.0.0/0              ctstate INVALID
ACCEPT     tcp  --  192.168.100.0/24       0.0.0.0/0              tcp dpt:80
ACCEPT     tcp  --  192.168.100.0/24       0.0.0.0/0              tcp dpt:8080
ACCEPT     tcp  --  192.168.100.0/24       0.0.0.0/0              tcp dpt:443
ACCEPT     tcp  --  192.168.100.0/24       192.168.200.3          tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0              192.168.200.3          tcp dpt:80
ACCEPT     tcp  --  192.168.100.3          192.168.200.3          tcp dpt:22
ACCEPT     udp  --  0.0.0.0/0              192.168.100.0/24       udp spt:53

Chain OUTPUT (policy DROP)
target     prot opt source                destination            tcp spt:22
ACCEPT     tcp  --  192.168.100.2          192.168.100.3
root@e5f5069a6631:/# █

```