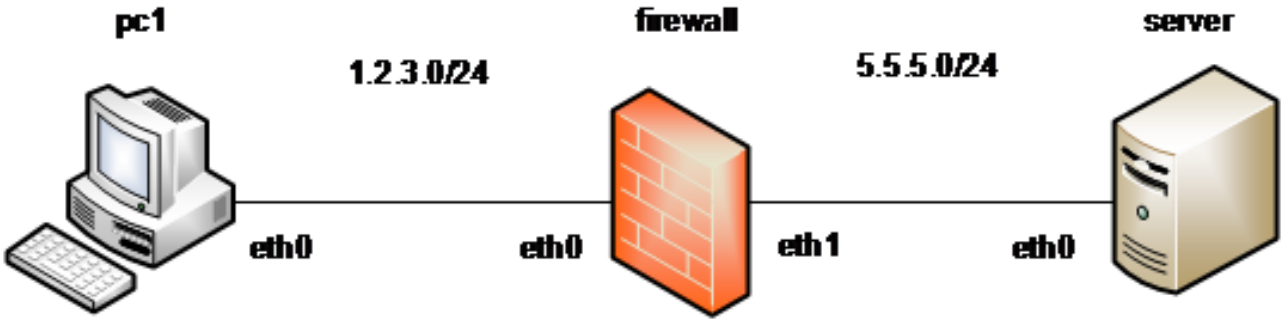
Sujet de TP4-INFO743

L'objectif de ce laboratoire est de mettre en place un serveur web utilisant Apache et un pare-feu simple utilisant iptables. La topologie du réseau est illustrée ci-dessous.



Introduction

Ce TP est structuré en trois parties. Tout d'abord, nous allons mettre en place un serveur web Apache afin de servir une simple page web. Ensuite, nous ajouterons un serveur de base de données SQL, qui sera utilisé par le serveur web. Enfin, nous mettrons en place un pare-feu pour contrôler l'accès à la page web.

Conseil : les commandes qui apportent des modifications et que vous souhaitez rendre permanentes peuvent être placées à la fin des fichiers de démarrage. Cela vous permet de reproduire rapidement le travail que vous avez effectué.

Pour tester les changements, vous pouvez créer de petits scripts Python/Bash, les placer dans le dossier /shared et les appeler à partir des fichiers de démarrage.

Exercice 1 -- Serveur web

Le serveur Apache diffuse des documents, par exemple des pages HTML, sur le web. L'exercice consiste à lancer le serveur Apache et à le configurer de manière à ce que, au niveau de l'application, certaines restrictions soient imposées aux documents accessibles.

Prenons maintenant quelques mesures pour configurer Apache.

1. Démarrez le laboratoire :

```
kathara lstart
```

2. Vérifier que le service apache2 fonctionne sur le serveur web.

```
/etc/init.d/apache2 status
```

Le dossier /etc/init.d contient les services d'une machine Linux en mode SysVInit. De nos jours, il est plus courant de trouver la commande systemctl en mode systemd plutôt qu'en mode SysVInit. Les deux sont des alternatives pour gérer l'initialisation de Les deux sont des alternatives pour gérer l'initialisation du système d'exploitation et de ses services.

3. Sur debian, les URL fournies par Apache renvoient par défaut à des fichiers du dossier `/var/www/html`. Dans le serveur web, exécutez la commande curl pour essayer de demander le fichier public/index.html.

```
curl -v 'http://127.0.0.1/public/index.html'
curl -v `http://127.0.0.1/notpublic/onlymine.jpg'
```

4. Après avoir effectué cette requête, remarquez qu'Apache stocke les informations relatives à la requête dans le fichier /var/log/apache2/access.log.

```
tail -n 10 /var/log/apache2/access.log
```

5. Apache contient différents modules qui peuvent être activés de manière facultative pour mettre en œuvre des fonctionnalités. L'un de ces modules est cgi (common gateway interface), qui permet d'exécuter des scripts lorsqu'un certain document est demandé. Pour activer le module :

```
a2enmod cgi
/etc/init.d/apache2 restart
```

6. Vous pouvez vérifier que le module a été activé en vérifiant que le dossier /etc/apache2/mods-enabled contient un fichier portant le nom du module (dans ce cas, cgi). Vous pouvez exécuter des scripts dans le dossier /usr/lib/cgi-bin/ en réponse à des requêtes HTTP, par exemple public/past_exam_answers.py :

```
curl -v 'http://127.1.2.3/cgi-bin/public/past_exam_answers.py'
```

7.En utilisant curl, remarquez comment, à partir de pc1, vous pouvez obtenir les pages web http:///public/index.html et cgi-bin/public/past_exam_answers.py comme prévu. Notez également que dans le fichier access.log, l'accès à partir de l'IP de pc1 a été enregistré.
Note : lorsque vous essayez d'accéder à past_exam_answers.py, vous obtiendrez une erreur car vous n'êtes pas connecté au service de base de données. Cette erreur sera corrigée dans l'exercice 2.

8. Remarquez qu'à partir de pc1, vous pouvez obtenir http:///notpublic/onlymine.jpg et cgi-bin/notpublic/future_exam_answers.py, ce qui ne devrait pas arriver. Au moins, nous avons pu vérifier que ces documents ont été consultés de manière incorrecte dans le fichier access.log. Nous devons corriger la configuration pour que les documents ne soient plus accessibles.

Partie avancée jusqu'à la fin de la question 1

9. Désactivons donc l'accès à ces documents. Pour ce faire, Apache vous permet de définir les permissions par dossier en créant un fichier .htaccess à la racine du dossier dans lequel vous souhaitez définir les permissions. Placez un fichier .htaccess avec ce contenu dans le dossier notpublic du répertoire du laboratoire et redémarrez le lab:

```
deny from all
```

10. Pour les fichiers du dossier /var/lib/cgi-bin, apache n'exécute pas les fichiers .htaccess par défaut. Pour y remédier, nous pouvons placer une règle équivalente à un fichier .htaccess dans le fichier /etc/apache2/apache2.conf. Nous pouvons rechercher des règles équivalentes dans le fichier /etc/apache2/apache2.conf et ajouter cette nouvelle règle :

```
<Directory /usr/lib/cgi-bin/notpublic/>
  deny from all
```

```
</Directory>
```
Alternativement, nous pouvons simplement activer les fichiers .htaccess pour le répertoire en question :
```
<Directory /usr/lib/cgi-bin/notpublic/>
  AllowOverride All
</Directory>
```
Redémarrez apache2 et relancez les requêtes avec curl et confirmez que la réponse du serveur est 403 Forbidden.

Comme nous utilisons une simulation, lorsque le laboratoire est redémarré, le fichier /etc/apache2/apache2.conf est effacé. Pour que ce changement persiste, vous pouvez ajouter les lignes suivantes au fichier webserver.startup (avant de démarrer apache) :

```
# Ajoutez ces 3 lignes
echo '' >> /etc/apache2/apache2.conf
echo ' deny from all' >> /etc/apache2/apache2.conf
echo '' >>
```

Si vous avez choisi d'utiliser AllowOverride All, modifiez-le en conséquence.

Exercice 2 -- Serveur de base de données

Maintenant que nous avons configuré le serveur Apache, ajoutons un nouveau serveur à notre réseau qui aura un service SQL (MariaDB), écoutant les requêtes sur le port 3306. Ce service sera utilisé par le serveur web sur la page http:///cgi-bin/public/past_exam_answers.py.

La machine sqlserver est déjà préconfigurée avec un serveur SQL. Il faut maintenant la connecter au réseau pour qu'elle soit accessible par le serveur. Vous devez ensuite modifier la deuxième ligne des fichiers past_exam_answers.py et future_exam_answers.py pour obtenir l'IP et le port d'accès au serveur sqlserver.

Procédez comme suit :

1. Regardez la topologie du réseau dans la figure, modifions-la pour que le serveur SQL soit également connecté au pare-feu.

2. Modifiez le fichier lab.conf pour connecter la machine sqlserver à un nouveau domaine de collision connecté au pare-feu.

3. Configurez le fichier firewall.startup pour établir la connexion à la nouvelle interface créée. Vous devrez peut-être modifier le sous-réseau configuré pour l'interface connectée au serveur web.

4. Confirmez que la commande suivante donne des résultats :
```
curl 'http:///cgi-bin/public/past_exam_answers.py'
```

5. Confirmez que la commande suivante ne donne aucun résultat :
```
curl 'http:///cgi-bin/notpublic/future_exam_answers.py'
```
Mais si vous l'exécutez localement sur le serveur web, elle devrait fonctionner :

```
python /usr/lib/cgi-bin/notpublic/future_exam_answers.py
```

Exercice 3 -- Parefeux

Maintenant que notre application web fonctionne correctement sur le serveur, examinons son comportement du point de vue de l'utilisateur. Rappelez-vous la topologie du réseau que vous avez utilisée dans l'exercice 2 et observons le comportement du pc1.

1.Sur le pc1, vérifiez que vous pouvez accéder aux examens passés et non aux examens futurs, comme sur le serveur :

```
curl 'http:///cgi-bin/public/past_exam_answers.py'
curl 'http:///cgi-bin/notpublic/future_exam_answers.py'
```

2.Malheureusement, nous avons encore des problèmes avec notre configuration réseau. Regardez ce qui se passe si vous exécutez la commande suivante sur le client :

```
nmap <ip du serveur>
```

3.Le port MariaDB (3306) est accessible au monde extérieur. Cela pose deux problèmes. Premièrement, il est possible pour un attaquant d'essayer de découvrir notre mot de passe (d'autant plus que le mot de passe est password, remplacez l'IP 5.5.5.3 par l'IP correspondant à sqlserver) :

```
# from pc1 :
mysql -u root -D sirs -e "SELECT * from exams ;" -h 5.5.5.3 -ppassword
```

Deuxièmement, même sans connaître le mot de passe, il est possible pour l'attaquant de "bombarder" le port de demandes de tentatives de connexion qui consomment des ressources sur le serveur et empêchent les connexions légitimes de s'établir.

4.Nous allons utiliser le pare-feu pour créer des règles qui nous obligent à n'exposer au monde extérieur que les services du réseau 5.5.5.0/24 que nous voulons explicitement exposer. Pour ce faire, nous utiliserons la commande iptables (sur debian10, nous devons utiliser la commande iptables-legacy, car actuellement la commande iptables ne fonctionne pas correctement).

5.Sur la machine du pare-feu, exécutez la commande suivante pour lister les règles actuelles du pare-feu :

```
iptables -L
```

Vous verrez qu'il existe 3 types de règles (INPUT, FORWARD, OUTPUT), appelées chaînes, et qu'aucune règle n'est actuellement configurée dans le pare-feu.

6.Si vous souhaitez supprimer le pare-feu, vous pouvez supprimer toutes les règles existantes en exécutant :

```
iptables -F
```

Commençons par ajouter une règle simple qui bloque tous les paquets ICMP (pings) destinés au pare-feu :

```
iptables -A INPUT -p icmp -j DROP
```

Si la commande échoue, n'oubliez pas d'utiliser la commande iptables-legacy au lieu d'iptables.

8. Observez le résultat sur le pare-feu (iptables -L) et vérifiez le résultat en essayant de faire un ping sur le pare-feu et le serveur web depuis pc1 et en établissant une connexion TCP (obtenir une page hébergée sur le serveur web).

```
ping 5.5.5.1
ping 5.5.5.2
```

```
curl 'http://5.5.5.2/public/index.html'
```
9. Supprimez la règle d'entrée que vous avez ajoutée :
```
iptables -D INPUT -p icmp -j DROP
```
10. Répétez les étapes 7 à 9, en remplaçant INPUT par FORWARD et OUTPUT, afin d'observer ce qui se passe dans chaque cas. Assurez-vous de bien comprendre ce que fait chaque règle. Pourquoi la chaîne FORWARD est-elle la plus utilisée dans un pare-feu ?

11.Si nous le souhaitons, nous pouvons également créer des règles plus spécifiques, par exemple, la règle suivante bloque les connexions TCP sur le port 23 depuis n'importe quelle adresse du réseau 42.42.0.0/16 provenant du réseau 192.168.2.0/24 (ou accepterait les connexions avec ces caractéristiques si nous changions DROP en ACCEPT).

```
iptables -A INPUT -p tcp -s 192.168.2.0/24 -d 42.42.0.0/16 --dport
23 -j DROP
```
12. Maintenant, sécurisons le réseau. Commencez par supprimer toutes les règles que vous avez créées dans le pare-feu :
```
iptables -F
```
13. Créez une ou plusieurs règles pour autoriser l'accès au serveur web, à savoir le port 80 de ce serveur utilisant le protocole TCP, sur lequel le protocole HTTP est encapsulé.

14.Créez une ou plusieurs règles pour bloquer tout autre trafic provenant de l'extérieur du réseau, c'est-à-dire passant par le pare-feu.

15. Testez que des connexions ne peuvent pas être créées sur d'autres ports en faisant en sorte qu'un service attende des connexions sur un port. Par exemple, vous pouvez utiliser la commande nc -l 1314 pour placer un processus en attente de connexions sur le port donné en argument.

16. Testez qu'aucun paquet provenant du monde extérieur ne peut accéder à sqlserver.

17. Testez qu'aucun paquet ICMP ne peut être envoyé de sqlserver vers le monde extérieur.

18.Listez les règles du pare-feu.

19. Notez que vous pouvez toujours recevoir des requêtes HTTP du pc1 vers le serveur web, mais que vous ne pouvez pas vous connecter au serveur sqlserver depuis le pc1.

Les objectifs d'isolation de la configuration du réseau ont été atteints !