SAE Sécurité informatique: Mise en place d'un Firewall et durcissement d'un serveur Web

IUT RT Rouen

1 Introduction

La sécurité informatique est un point crucial pour toutes les entreprises aujourd'hui. Parmi les outils à utiliser, le concept de *Firewall* est un élément clé de toute architecture réseau. Mais les serveurs web sont aussi une cible privilégiée pour les attaques.

Dans ce projet, vous vous intéresserez à la mise en place d'un firewall et vous mettrez en évidence son efficacité avec des démonstrations de son fonctionnement. Vous aborderez également la sécurisation du serveur web Nginx [1].

1.1 Prérequis

Mener ce projet à bien impliquera un grosse partie de documentation personnelle.

Il faudra également s'interesser aux outils de test de pénétration (dans un cadre "*white hat*" [2]), afin de pouvoir vérifier l'efficacité des mesures techniques que vous allez mettre en place ("sniffage" de port, de réseau). On peut citer : nmap, wireshark, Angry IP Scanner, ...

Ce document inclus un grand nombre de sources, qu'il faudra explorer, mais il pourra être nécessaire d'en trouver d'autres.

2 Description

2.1 Cahier des charges

Vous êtes responsable informatique dans la PME XXX, et votre dirigeant vous demande de réaliser une étude de la sécurisation du réseau sous deux angles :

— mise en place d'un firewall

— securisation du serveur web de l'entreprise, qui est hébergé en local

Il vous demande de lui faire une démonstration de la sécurité apportée par un Firewall, en lui montrant des exemples avant/après, démontrant que certaines choses qui étaient possible sans le Firewall sont maintenant interdites par celui-ci, tout en permettant les connexions entrantes et sortantes "légitimes".

De même, et de façon indépendante du point précédent, vous devez être capable de montrer avec une démo que votre sécurisation du serveur Web a permis de diminuer la surface d'attaque.

Il vous demande également de faire la démonstration de la mise en place d'une "DMZ" avec le Firewall, afin de protéger le réseau local des intrusions.

1. Cette partie sera probablement la plus simple du projet.
2. `https://fr.wikipedia.org/wiki/White_hat`

2.2 Modalités pratiques

— Pour la démonstration, vous travaillerez avec deux VM Debian, configurées et pilotées avec Vagrant.
  — l'une qui jouera le role de serveur, sur laquelle sera implantée le serveur web nginx (qui servira une simple page statique) et le firewall ;
  — l'autre qui servira pour les tests, à la fois pour vérifier que l'accès aux pages web est possible, mais aussi pour vérifier que des attaques classiques sont contrées.

  En fin de projet, il en faudra une 3è, pour la demonstration de la DMZ.

— Travail en équipe de deux, constitution des équipes libre (une seule équipe de trois, dans le groupe à effectif impair).

— Durée projet : deux semaines,

— Il vous est demandé un dépot Github nommé **`demo-firewall`**, dans lequel vous versionnerez vos fichiers, tests compris.

— Livrables :
  — Scripts de constructions des VM
  — Scripts de démonstration du fonctionnement : arret/démarrage du Firewall + commandes de connexion (qui échouent ou pas selon l'état du Firewall)
  — Traces des étapes (journal)

  Une soutenance orale du projet en fin de projet.

  L'ensemble de ces éléments seront à prévoir sur le dépôt GH, sous forme de "release" [3]

3 Introductions aux *firewalls*

L'objectif d'un firewall est de filtrer (=autoriser ou interdire) les flux de données entrantes et sortantes, à partir d'un ensemble de règles. Il s'agit de sécuriser le réseau local d'une entreprise tout en permettant l'accès à l'extérieur, et permettre également les connexions entrantes, moyennant certaines contraintes.

3.1 Types de firewalls

On distingue plusieurs niveaux (voir Fig. 1) :

— Firewall "stateless" (sans état) : les règles sont indépendantes de l'historique, chaque nouvelle connection est évaluée indépendamment de la connection précédente, via des règles (ACL) basées sur les adresses, les ports, et les protocoles.

— Firewall "statefull" (avec état) : les connections sont évaluées en fonctions de l'historique. Par exemple si une machine du réseau local a sollicité un service spécifique à l'exterieur, alors le firewall pourra autoriser une connexion en retour de la même IP, même si le port est à l'origine interdit.

Certains Firewalls peuvent aussi permettre de mettre en place une "DMZ" [4], en routant sur un réseau spécifique certaines connexions (voir Fig. 2).

3. Il faut préalablement créer un "tag" Git, le pousser sur le dépot distant, pour ensuite créer sur l'interface web GH une "release", qu'on rattache au "tag" crée.

4. Voir `https://fr.wikipedia.org/wiki/Zone_d%C3%A9militaris%C3%A9e_(informatique)` et `https://www.youtube.com/watch?v=mY-TvNXFHl0`

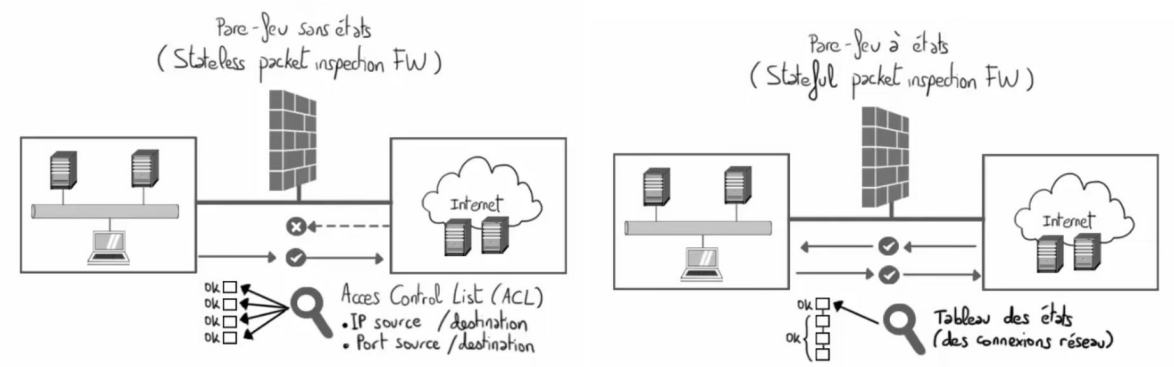

FIGURE 1 – Source : "Cookie Connecté", https://www.youtube.com/watch?v=6Swt51w3EjY

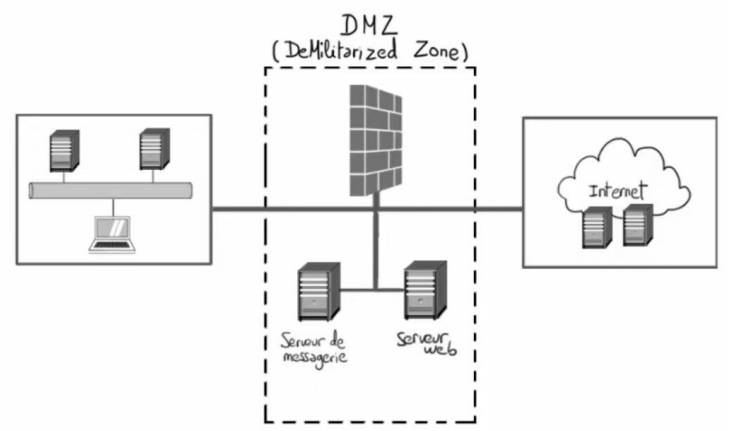

FIGURE 2 – Source : "Cookie Connecté", https://www.youtube.com/watch?v=6Swt51w3EjY

3.2 Implémentations

Le firewall le plus courant dans le monde Linux est **iptables**, qui interagit avec le "framework" interne au kernel **netfilter**[5]. Pour bien en comprendre les concepts, prendre le temps de lire cet article jusqu'à la section 1.2 incluse :
https://connect.ed-diamond.com/GNU-Linux-Magazine/glmfhs-041/introduction-

Voir aussi ces sources sur la collaboration entre **netfilter** et **iptables** :

— https://en.wikipedia.org/wiki/Netfilter

— https://blog.cloudsigma.com/the-architecture-of-iptables-and-netfilter

— https://www.digitalocean.com/community/tutorials/a-deep-dive-into-ipta

L'outil **iptables** est en voie de remplacement dans les distributions majeures par l'outil **nftables**, mais pendant le temps de la transition, on rencontre fréquemment les deux. Vous trouverez donc des articles décrivant les deux, mais il est évidemment préférable de ne pas activer les deux en même temps.

Par ailleurs, **iptables** étant assez complexe à configurer, on peut utiliser sur Debian/Ubuntu l'outil **ufw** (*Uncomplicated FireWall*) qui est en réalité un "*front-end*" par dessus **iptables** permettant d'en simplifier la mise en place. On peut trouver une courte introduction ici :
https://doc.ubuntu-fr.org/ufw

Dans un premier temps, ce sera l'outil à utiliser pour bien se familiariser avec la configuration d'un Firewall.

Ressources / Firewall :

— https://pled.fr/?p=19054

— https://linux-audit.com/nftables-beginners-guide-to-traffic-filtering/

— https://www.linode.com/docs/guides/how-to-use-nftables/

— https://www.it-connect.fr/cours/filtrage-reseau-et-pare-feu-avec-netfi

3.3 Evaluation de la performance

Pour tester votre VM équipée de son Firewall, de multiples outils sont disponibles. Le plus universel est **nmap**, qui est un "scanneur" de ports très puissant, mais on peut en envisager d'autres.

Pour évaluer la capacité du serveur web à répondre à de fortes charges, vous pouvez utiliser les outils **hey** (**sudo apt install hey**, puis **man hey**) ou **salvo** (**pip install salvo**).

Quelques sources :

— https://www.cyberciti.biz/faq/unix-linux-check-if-port-is-in-use-comma

— https://connect.ed-diamond.com/linux-pratique/lp-129/explorez-votre-re

— La référence complete : https://nmap.org/book/toc.html et notamment le chap. 10 : https://nmap.org/book/firewalls.html

5. https://www.netfilter.org/

4 Sécurisation de serveur Web

Le serveur Nginx offre une flexibilité d'usages très large (proxy, load-balancer, serveur web statique, dynamique, ...) mais sa configuration par défaut peut laisser de nombreuses failles de sécurité ouvertes.

Il est donc prudent d'explorer ses possibilités et de voir quelles améliorations on peut apporter à la configuration par défaut.

Note : parmi les recommandations, certaines suggèrent la recompilation de l'exécutable à partir des sources, en paramétrant la compilation pour inclure/exclure certaines fonctionnalités. Cet aspect sort clairement du cadre de ce projet, ne pas se lancer sur cette voie.

4.1 Sources

— `https://www.tecmint.com/nginx-web-server-security-hardening-and-perfor`
— `https://blog.jbriault.fr/security-nginx/`
— SSL/TLS "Best Practices" : `https://github.com/ssllabs/research/wiki/SSL-and-TL`

5 Etapes du travail

Le travail est assez complexe. Afin de conserver l'approche IaC, il vous est demandé de travailler de façon **scriptée**, en construisant les VM du projet avec Vagrant. Il faudra donc se plonger en profondeur dans les possibilités de configuration d'une VM construite avec Vagrant, au delà de ce que vous avez vu en TP. Il sera pertinent de relire les slides du cours sur Vagrant [6].

Il faudra s'intéresser aux concepts des firewall et comprendre les règles qu'on peut y placer, et comment on peut vérifier qu'elles sont fonctionnelles.

Dans un premier temps, vous devrez faire des essais dans une VM en utilisant l'outil **`ufw`** "à la main", et comprendre comment tester la réponse du firewall. Il faudra se familiariser avec les outils curl, nmap, ...

La distribution du travail au sein de l'équipe et la tracabilité de votre travail seront des points clés. Vous maintiendrez sur votre dépot un fichier journal (similaire à celui utilisé pour le projet précédent), que vous complèterez en **fin de chaque séance** en autonomie.

Ce journal devra préciser ce qui a été validé lors de la séance, et les points qui posent des difficultés.

Cette communication permettra de collecter des **traces** du travail, que vous pourrez ensuite valoriser.

5.1 Démarrage du projet

Dans un dépot git cloné sur votre machine, et dans un dossier de travail **`sae_Firewall`**, commencer par créer deux dossiers **`server`** et **`client`**, et dans chacun d'eux, y créer une machine Debian 12 avec Vagrant [7], afin d'avoir un "Vagrantfile" initial. Vérifier que les machines démarrent correctement, puis les arreter.

Editer ensuite le Vagrantfile du serveur pour y placer le nécessaire pour installer Nginx (ainsi que le paquet **`nginx-extras`**), et le firewall **`ufw`**.

6. `https://universitice.univ-rouen.fr/mod/resource/view.php?id=1117559`
7. La "box" debian/bookworm64 (Debian 12) est tout à fait adaptée

Après redémarrage des deux VM, prendre la main dans le client, et y faire des tests : pouvez vous "pinger" le serveur ? Renommer les VM et les placer sur un réseau privé en ajoutant dans la configuration :

```
  config.vm.hostname = "server" # (ou client)
# mettre deux IP différentes à la place de XX sur serveur et client:
  config.vm.network :private_network, ip: "192.168.0.XX"
```

Il faudra évidemment installer au fur et à mesure installer des paquets manuellement dans chacune des machines. Noter soigneusement les paquets que vous installez "à la main", afin de pouvoir ensuite ajouter ceci dans les "Vagranfile" des deux machines.

Copier dans la machine serveur un fichier de configuration de nginx qui va lui faire servir une page statique élémentaire, et vérifier ensuite dans le client avec curl que vous pouvez y accéder.

Une fois la configuration de vos deux VM stabilisée et les VM fonctionnelles, l'étape suivante consiste à étudier en deux phases les aspects "sécurité" [8] :

— documentez vous sur les failles possible de Nginx, et essayer de les démontrer. Puis, au fur et à mesure, préparer un script ou un nouveau fichier de configuration et montrer qu'avec une nouvelle configuration, la faille n'apparait plus. Il doit être possible de basculer du mode "standard" au mode "plus sécurisé" simplement avec le lancement d'un script.

— mettez en évidence de façon expérimentale (par essais manuels) les failles de sécurités liées à l'absence d'un Firewall, puis recensez les et préparer un script exécutable vous permettant d'en faire la démo à votre dirigeant.

  Puis, de la même façon, commencer à activer de façon manuelle [9] certaines règles, et vérifier qu'elles sont opérationnelles.

  Une fois vos expérimentations stabilisées, rassembler ces commandes de configuration dans un script qui sera exécuté au démarrage de la VM serveur. Et rassembler les commandes de test d'intrusion dans un script qui pourra être exécuté sur le client.

5.2 Notes sur Vagrant

Lorsque vous arretez la VM, penser à la relancer avec l'option permettant de recharger la configuration : `$ vagrant up --provision`
Ou, si la machine est toujours "running", vous pouvez faire `$ vagrant reload`

8. Ces deux phases pourront être traitées séparément par les deux membres de l'équipe.
9. Via la commande **ufw**