

HOME SERVER

Étienne Gauvin-Clermont (18 094 321)

Michael Labrecque (18 135 643)

Mise en situation

De nos jours, il devient de plus en plus intéressant de vouloir se confectionner un serveur maison (home server), et ce, pour plusieurs raisons. Pour en citer quelques une :

- Centralise et fournit un contrôle absolu sur les données et traitements
- Hybride : peut faire d'autres tâches que son objectif premier

Exemple

Prenons un home server dont son principal objectif est le stockage d'informations. On peut lui demander parallèlement d'exécuter d'autres tâches. En effet, nous pouvons lui demander par exemple d'encoder des vidéos avec ffmpeg, mettre en place un serveur pour un jeu, faire de l'interpolation de vidéo avec [RIFE](#).

- Pleine gestion sur la disponibilité du serveur
- Gestion des sauvegardes
- **Potentiel** maintien de la privacité des données

Dans la situation où un serveur est utilisé uniquement à des fins personnelles et de façon locale, il est tout à fait acceptable de limiter son implication dans le but de sécuriser pleinement celui-ci. Toutefois, il peut être intéressant de rendre accessible notre serveur de l'extérieur (remote access). Ainsi, il serait désormais possible de consulter et interagir avec notre serveur de n'importe quelle endroit. Pour ce faire, il suffit de port forward le serveur. Une telle configuration est très triviale, mais elle à un problème.

Le problème est que port forward notre serveur nous rend plus propice aux attaques. Effectivement, rendre notre système interrogeable de l'extérieure pour nous le rend aussi pour les autres. Pour cette raison, nous devons mettre certains concepts en place pour limiter le plus possible une future tentative d'intrusion.

Note

Il y a plusieurs façons de sécuriser un home server. Nous couvrirons uniquement ceux que nous jugeons pertinents et intéressants dans le cadre du cours. Aussi, les pratiques mises de l'avant diminuent le risque d'intrusion sans pour autant les éliminer à 100%.

Ce document va illustrer certaines configurations que nous avons testées et mises en place. D'ailleurs, certaines de ces configurations ont été automatisées dans un projet typescript que nous avons développé et mis disponible sur [GitHub](#). Son comportement reste limité, mais ça structure est faite de sorte à être très propice à l'ajout de configuration.

Modèle de sécurité

Sachant que notre serveur peut être accessible ou interrogé par n'importe qui sur internet, nous avons l'obligation de mettre en place certaines mesures de sécurité afin de protéger nos données.

Activer firewall

Prérequis

```
sudo apt install ufw
```

Pour activer le firewall, exécuter:

```
sudo ufw enable
```

Ensuite, nous devons autoriser le port SSH (par défaut 22) dans le firewall en exécutant:

```
sudo ufw allow 22/tcp
```

Note

Il est parfois nécessaire d'incorporer un fichier de configuration dans `/etc/ufw/applications.d` pour ajouter des règles détaillées.

Exemple de règle pour un [Plex](#)

```
[plexmediaserver]
title=Plex Media Server (Standard)
description=The Plex Media Server
ports=32400/tcp|3005/tcp|5353/udp|8324/tcp|32410:32414/udp

[plexmediaserver-dlna]
title=Plex Media Server (DLNA)
description=The Plex Media Server (additional DLNA capability only)
ports=1900/udp|32469/tcp

[plexmediaserver-all]
title=Plex Media Server (Standard + DLNA)
description=The Plex Media Server (with additional DLNA capability)
ports=32400/tcp|3005/tcp|5353/udp|8324/tcp|32410:32414/udp|1900/udp|32469/tcp
```

Note

C'est une bonne pratique de séparer les règles dans plusieurs fichiers. Par exemple, les règles précédentes pourraient être contenues dans un fichier

`PlexMediaServer`

Ensuite, il suffit d'activer certaines règles par leur nom. Par exemple:

```
sudo allow plexmediaserver-all
```

Note

Un *reload* est parfois nécessaire pour prendre en considération certaines modifications. `sudo ufw reload`

Désactiver le root login

L'une des premières étapes dans la sécurisation d'un serveur est de désactiver l'accès à celui-ci de l'extérieur avec le compte de super utilisateur. Comme l'une des méthodes d'accès via un terminal est SSH, nous devons d'abord désactiver la connexion à la machine depuis l'utilisateur root.

Quelques raisons pour lesquelles il est préférable de désactiver cet utilisateur :

- Il possède énormément trop de droits sur la machine. (Il possède tous les droits)
- Même avec un mot de passe relativement fort il reste vulnérable à une attaque par force brute.

Étapes

1. Ouvrir le fichier de configuration `sshd_config`

```
nano /etc/ssh/sshd_config
```

2. Chercher la ligne `PermitRootLogin yes` et remplacer la valeur `yes` par `no`
3. Sauvegarder le fichier
4. Redémarrer le service SSH

```
service ssh restart
```

Utiliser clef SSH plutôt qu'un mot de passe

Comme nous allons nous connecter à notre serveur exposé depuis un réseau qui n'est pas sécurisé, l'utilisation d'une clé Secure shell nous permettra de créer une connexion sécurisée. La clé SSH utilise de la cryptographie à clé publique. C'est un algorithme composé de deux clés. Une clé publique mise à la disponibilité de tous et une clé privée bien gardée sur la machine du client qui sert à déchiffrer les messages chiffrés à l'aide de la clé publique correspondante.

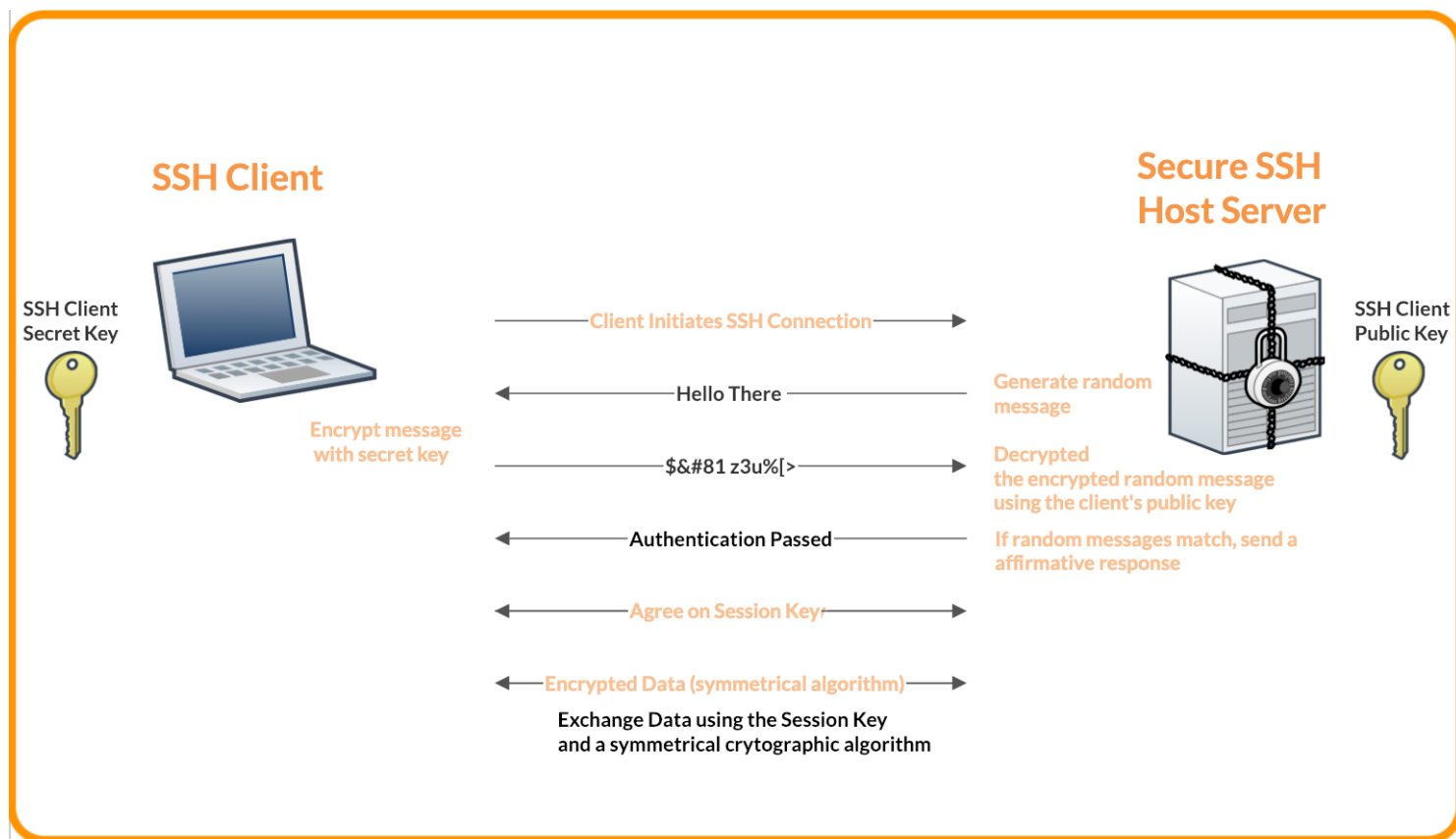
Il existe plusieurs algorithmes permettant de créer une paire de clés SSH :

DSA	RSA	Ed25519
Dangereux et plus utilisé depuis la version 7.0 de OpenSSH	Offrant une sécurité acceptable si la longueur de la clé est de 3072 ou 4096 bits	L'algorithme à privilégier de nos jours, car actuellement le plus sûr.

L'utilisation d'une phrase secrète pour protéger la clé privée est une excellente façon de s'assurer que nous sommes la seule personne qui puisse l'utiliser principalement si nous ne sommes pas le seul utilisateur de la machine. De plus, cela ajoute une étape supplémentaire à un attaquant qui réussirait à prendre possession de la clé privée puisqu'il devra faire une attaque par force brute afin de trouver cette phrase avant d'utiliser la clé privée.

Afin de s'authentifier au serveur avec sa clé ssh, le client procède comme suit:

1. Le client initialise la connexion ssh
2. Le serveur génère un message aléatoire
3. Le client encrypte le message aléatoire avec sa clé privée
4. Le serveur décrypte le message encrypté avec la clé publique du client
5. Dans le cas où le message décrypté est le même que celui initialement envoyé au client, le serveur envoie une réponse affirmative au client
6. Le client et le serveur s'entendent sur l'utilisation d'une clé de session
7. Échange de données encrypté avec un algorithme symétrique et la clé de session



Étapes

Créer clef client

Afin de générer une clé, il suffit de procéder comme suit :

1. Ouvrir un terminal
2. Générer la clé avec la commande
 - Pour Ed25519

```
ssh-keygen -t ed25519
```
 - Pour RSA

```
ssh-keygen -t rsa -b 4096
```
3. Laisser l'emplacement par défaut
4. Entrez une phrase secrète pour protéger la clé privée

Cela peut être intéressant d'avoir plusieurs clefs ssh, car on peut plus facilement gérer les restrictions et l'utilisation de ceux-ci. En effet, il pourrait arriver la situation où nous voulons désactiver/invalider une certaine clef en plus de savoir exactement quelle clef a été utilisée à quel moment.

Pour créer une clef RSA de 4096 bits portant le nom de client, exécuter:

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/client-key
```

Note

Le nom *client* aurait pu être remplacé par n'importe quoi.

Il sera proposé lors de la création de la clef de fournir un passphrase. Il est fortement conseillé de fournir un passphrase, sans quoi seule la possession de la clef privée (client-key) sera nécessaire pour l'authentification.

Après, le dossier `/home/username/.ssh` contient les fichiers suivants :

- client-key
- client-key.pub

Résultat

```
etienne@pop-os:~$ ssh-keygen -t rsa -b 4096 -f ~/.ssh/client-key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/etienne/.ssh/client-key
Your public key has been saved in /home/etienne/.ssh/client-key.pub
The key fingerprint is:
SHA256:l2W9aDIC1j1bkYIToL36bekVICfbsV2+AM2E1Ance5g etienne@pop-os
The key's randomart image is:
+---[RSA 4096]---+
|      ...000*0.  |
|      0 .00.0=*  |
|      . + 0.*0=0= .|
|      . 0 0 @.E = |
|      . S 0 =.+ . |
|      . 0 = . .  |
|      . . .      |
|      . .0.      |
|      .00        |
+-----[SHA256]-----+
```

Étapes

Transférer clef client au serveur

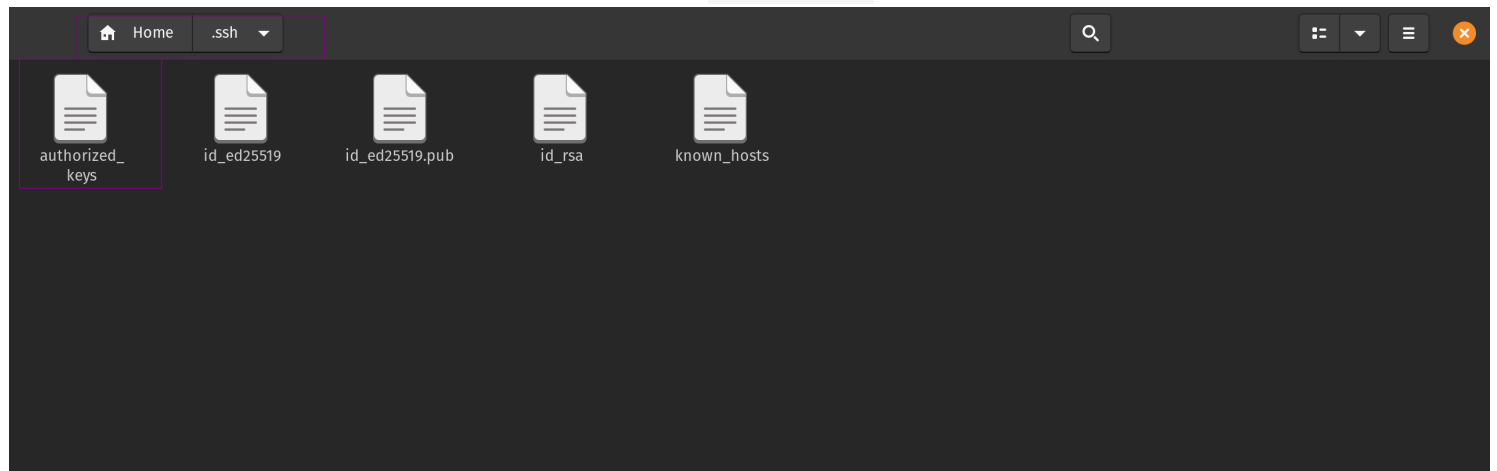
Pour copier la clef publique du client vers le serveur, exécuter:

```
cat ~/.ssh/client-key.pub | ssh server_username@server_ip "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

Note

Remplacer *server_username* avec le nom du serveur et remplacer *server_ip* par l'ip du serveur.

Maintenant, le dossier du serveur `/home/username/.ssh` contient le fichier `authorized_keys`



Étapes

Connexion au serveur

Pour se connecter au serveur, exécuter:

```
ssh -i ~/.ssh/client-key server_username@server_ip
```

Note

Remplacer *server_username* avec le nom du serveur et remplacer *server_ip* par l'ip du serveur.

Ensuite, il faudra saisir le passphrase entré dans les étapes précédentes.

```
etienne@pop-os:~$ ssh -i ~/.ssh/client-key etienne@10.0.0.33
Enter passphrase for key '/home/etienne/.ssh/client-key':
Welcome to Pop!_OS 20.04 LTS (GNU/Linux 5.15.5-76051505-generic x86_64)

* Homepage: https://pop.system76.com
* Support: https://support.system76.com

Last login: Thu Dec 9 08:51:12 2021 from 10.0.0.243
```

Toutefois, nous sommes toujours en mesure de nous connecter au serveur par le biais des informations d'un utilisateur. En effet, l'ajout d'une clef publique au serveur n'empêche pas automatiquement l'authentification par mot de passe.

```
etienne@pop-os:~$ ssh etienne@10.0.0.33 Server
etienne@10.0.0.33's password:
Welcome to Pop!_OS 20.04 LTS (GNU/Linux 5.15.5-76051505-generic x86_64)

* Homepage: https://pop.system76.com
* Support: https://support.system76.com

Last login: Thu Dec 9 08:51:03 2021 from 10.0.0.243
```

Authentification uniquement par la clef SSH

Il est maintenant primordial de vouloir empêcher toute authentification avec le mot de passe de l'utilisateur. Pour refuser une connexion ssh utilisant ce principe, il suffit de modifier le fichier `/etc/ssh/sshd_config` pour y ajouter cette ligne :

```
PasswordAuthentication no
```

Ensuite, il sera impossible de se connecter avec un mot de passe d'utilisateur.

```
etienne@pop-os:~$ ssh etienne@10.0.0.33
etienne@10.0.0.33: Permission denied (publickey).
```

Ajouter un système MFA à la connexion SSH

L'utilisation d'un mot de passe ou la clé SSH sont 2 facteurs d'authentification différents. Un facteur d'authentification est une information utilisée afin de prouver que l'utilisateur a le droit d'effectuer des actions sur un système tel que de s'y connecter. Ces facteurs utilisent différents canaux d'authentification tels qu'un ordinateur, un téléphone cellulaire ou une clé physique d'authentification. C'est le média qui sert à transmettre un facteur d'authentification à l'utilisateur. Afin de renforcer la connexion ssh à notre serveur, c'est une bonne pratique d'appliquer une sécurité supplémentaire par l'utilisation de MFA (Multi-factor authentication). De cette façon, un attaquant qui réussit à compromettre votre ordinateur de bureau doit aussi obtenir le contrôle d'un ou plusieurs autres appareils vous appartenant afin de pouvoir effectuer ses actions malicieuses.

Les types de facteurs se catégorisent en 3 groupes:

- Quelque chose que vous connaissez: un mot de passe ou une question de sécurité
- Quelque chose que vous possédez: une application d'authentification ou un token de sécurité.
- Quelque chose propre à vous: une empreinte digitale, ou la reconnaissance vocale

L'un des facteurs fréquemment utilisés par les différents systèmes est une application OATH_TOTP (Open Authentication Time-Based One-Time Password). OATH_TOTP est un protocole utilisant un mot de passe généralement composé de 6 à 8 caractères utilisables une seule fois et qui se rafraîchit après une période de temps d'environ 30 secondes. Un exemple de ces applications serait Google authenticator ou Microsoft authenticator. Un tel système peut être configuré de la façon suivante.

Étapes

Installer Google PAM

- Installer l'application

```
sudo apt install libpam-google-authenticator
```
- Lancer l'application

```
google-authenticator
```
- Répondre 'y' à la question `Do you want authentication tokens to be time-based (y/n)`
- Répondre 'y' à la question

```
Do you want to disallow multiple uses of the same authentication token? This restricts you to one login about every 30s, but it increases security. (y/n)
```
- Répondre 'n' à la question

```
By default, tokens are good for 30 seconds and in order to compensate for possible time-skew between the client and the server, we recommend you to set up a time-based one-time password (TOTP) application. (y/n)
```
- Répondre 'y' à la question

```
If the computer that you are logging into isn't hardened against brute-force login attempts, you can enable rate-limiting for the authenticator. (y/n)
```

Étapes

Configurer Pam pour OpenSSH

- Ouvrir la configuration sshd avec votre éditeur préféré

```
sudo nano /etc/pam.d/sshd
```
- Commenter la ligne `include common-auth` au début du fichier avec le caractère #

Note

Cela informe PAM de ne pas demander pour un mot de passe, mais seulement pour la vérification du second facteur. Laisser la ligne non commentée utilise 3 facteurs (la clé ssh, le mot de passe du compte, la vérification du TOTP)

- Ajouter la ligne `auth required pam_google_authenticator.so nullok` à la toute fin du fichier

Note

nullok signifie que la méthode d'authentification PAM est optionnel. Cela permet à un utilisateur sans jeton OATH-TOTP de se tout de même se connecter via ssh.

Important: Retirer la valeur nullok lorsque tous les utilisateurs ont configuré leur jeton OATH-TOTP

- Sauvegarder et fermer

Étapes

Configurer ssh pour qu'il gère l'authentification PAM

- ouvrir le fichier `/etc/ssh/sshd_config` dans votre éditeur préféré
`sudo nano /etc/ssh/sshd_config`
- Trouver le paramètre `ChallengeResponseAuthentication` et assigner la valeur yes
- Ajouter la ligne suivante
`AuthenticationMethods publickey,password publickey,keyboard-interactive`
- Sauvegarder et fermer le fichier
- Redémarrer le service sshd
`sudo systemctl restart sshd.service`

Configuré un intervalle de délais d'inactivité

Cela permet de fermer une connexion ssh qui n'est plus utilisé. Cela évite de consommer inutilement des ressources système tout en évitant qu'une personne tierce puisse accéder à la connexion en accédant à l'ordinateur de l'utilisateur connecté. Pour cela, il suffit de :

1. Ouvrir le fichier `sshd_config` avec votre éditeur préféré
`sudo nano /etc/ssh/sshd_config`
2. Décommenter et attribué la valeur désirée en seconde 300 (5min) à la variable **ClientAliveInterval**
3. Sauvegarder et fermer
4. Relancer le service
`sudo systemctl restart sshd.service`

Changer le port SSH par un personnalisé

Le port utilisé par défaut est le 22. Cette valeur est largement connue au sein des attaquants. Les différents outils de sécurité et d'intrusion utilisent donc principalement ce port. Même si le modifier revient à faire de la sécurité par l'ignorance qui est très faible comme sécurité, cela permettra néanmoins de réduire considérablement les attaques automatisées sur le port SSH

1. Ouvrir le fichier `sshd_config` avec votre éditeur préféré
`sudo nano /etc/ssh/sshd_config`
2. Décommenter et attribué la valeur désirer ex. 500 à la variable **Port**
3. Sauvegarder et fermer
4. Relancer le service
`sudo systemctl restart sshd.service`

Note

Ne pas oublier de modifier le transfert des ports sur le routeur pour la même valeur.

Désactivé le transfert X11

Premièrement, c'est inutile pour un serveur qui est dans 99.9% des cas utilisé avec un terminal. Ensuite, cela empêchera certains vecteurs d'attaque qui exploite le transfert X11 qui n'a pas été conçu en ayant la sécurité comme point fort lors du développement. Le fait qu'il soit activé n'est pas mauvais dans le cas où le serveur n'est pas compromis. Le transfert X11 amène une plus grande confiance du côté serveur qu'à celui du client. S'il a été compromis, il peut servir à effectuer une multitude d'actions sur la machine du client tel qu'ouvrir ou fermer des fenêtres et espionner les touches saisies et injecter des événements clavier ou souris. En le désactivant, nous évitons un potentiel vecteur de propagation d'un malware sur la machine client depuis notre serveur.

Pour cela, il suffit de :

1. Ouvrir le fichier `sshd_config` avec votre éditeur préféré

```
sudo nano /etc/ssh/sshd_config
```

2. Changer la valeur pour **no** à la variable **X11Forwarding**
3. Sauvegarder et fermer
4. Relancer le service

```
sudo systemctl restart sshd.service
```

Attaque pouvant être menée sur un home server

Tous serveurs peuvent être la cible d'attaque principalement lorsque les services sont exposés sur le web. Parmi celles-ci on peut retrouver les attaques Man-in-the-Middle, le déni de service, brute force des identifiants du port ssh. Certaines sont plus faciles à contrer que d'autres.

Man-in-the-Middle

Cette attaque, comme son nom l'indique, met en place un acteur malveillant entre nous et la machine/service que l'on veut joindre. Le but est d'intercepter les communications échangées entre les deux entités sans que sa présence ne soit remarquée. Dans ce type d'attaque, l'homme du milieu peut lire, mais aussi modifier les messages échangés. Le protocole de la connexion SSH avec une clé ssh permet de se prémunir contre cette attaque. En effet, dans le cas où un utilisateur nommé Ève tenterait de se mettre entre l'utilisateur Alice et le serveur de Bob, Ève pourrait tenter d'initier la connexion avec Bob à la place d'Alice et se faire passer pour le serveur de Bob auprès d'Alice. Cependant, Ève ne réussirait pas à joindre le serveur de Bob puisqu'elle ne réussirait pas à passer l'étape de l'encryption du mot généré aléatoirement [à l'étape 2](#), car sa clé ssh public n'est pas sauvegardé dans le serveur de Bob. Ainsi, lorsque Bob tenterait de décoder le message reçu par Ève, celui-ci ne correspondrait pas au mot qu'il a initialement envoyé et la connexion serait abandonnée.

Attaque par déni de service (DOS)

C'est une attaque qui vise à rendre un système indisponible et elle peut prendre différentes formes. Par exemple, la saturation de la bande passante qui rend ainsi le serveur injoignable et même l'épuisement des ressources système de la machine qui l'empêche de répondre au trafic légitime. Le déni de service opère par l'envoi d'une multitude de requêtes rapidement sur la machine visée afin de rendre le service instable voir indisponible. Cette attaque est la plus difficile à mitiger pour un home serveur puisque même si des règles de pare-feu sont appliquées et bloquent les connexions malicieuses. Les demandes de connexion peuvent tout de même arriver en grande quantité et saturer la bande passante. La solution la plus probable dans ce cas est que le fournisseur d'accès internet bloque les requêtes faites vers le home server. Plusieurs services existent afin de mitiger au maximum ce type d'attaque. Parmi eux on trouve SSHGuard, Fail2ban and DenyHosts. Aucun d'eux n'a été testé ici.

Brute force SSH

Prérequis

```
sudo apt install ipcalc
sudo apt install nmap
sudo apt-get install -y hydra
```

Note

Les paquets précédents doivent être installés pour réaliser la totalité du brute force.

Comme nous avons mentionné, il est tout à fait réaliste d'imaginer une situation où des tentatives de connexions seront faites sur notre serveur pour causer préjudice. Cette section va tenter de mettre en évidence l'importance des précautions proposées dans les sections précédentes.

Note

Nous allons faire abstraction sur les moyens utilisés par un individu pour se connecter avec succès au network qui occupe notre machine qui sera brute force.

Sachant qu'une personne mal intentionnée réussie à ce connecter au network, celui-ci peut tenter de brute force les machines ayant un service ssh. Pour réussir ce tour de force, il va falloir qu'il trouve l'ip de toutes les machines ayant un service ssh activer en effectuant un network map scan. Pour effectu   un network map scan, il faut d'abord avoir l'intervalle du network.

Obtenir network informations

Pour obtenir cet intervalle, il faut d'abord connaitre l'ip courant de la machine en ex  cutant:

```
ifconfig | grep inet
```

R  sultat

```
etienne@pop-os:~$ ifconfig | grep inet
inet 172.18.0.1 netmask 255.255.0.0 broadcast 172.18.255.255
inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
inet 10.0.0.243 netmask 255.255.255.0 broadcast 10.0.0.255
inet6 fe80::d789:e621:f8ca:e884 prefixlen 64 scopeid 0x20<link>
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
```

Donn  es recueillies

IP courant
10.0.0.243

Maintenant, nous pouvons obtenir    partir de l'adresse ip l'intervalle du network en ex  cutant:

```
ipcalc 10.0.0.243
```

R  sultat

```
etienne@pop-os:~$ ipcalc 10.0.0.243
Address: 10.0.0.243          00001010.00000000.00000000. 11110011
Netmask: 255.255.255.0 = 24  11111111.11111111.11111111. 00000000
Wildcard: 0.0.0.255         00000000.00000000.00000000. 11111111
=>
Network: 10.0.0.0/24        00001010.00000000.00000000. 00000000
HostMin: 10.0.0.1          00001010.00000000.00000000. 00000001
HostMax: 10.0.0.254        00001010.00000000.00000000. 11111110
Broadcast: 10.0.0.255      00001010.00000000.00000000. 11111111
Hosts/Net: 254              Class A, Private Internet
```

Donn  es recueillies

IP courant	Intervalle network
10.0.0.243	10.0.0.0/24

Scan le network

Il est d  sormais possible de scanner tous les appareils se retrouvant dans le network ayant le port ssh utilis   (22) et ouvert.

```
sudo nmap 10.0.0.0/24 -p 22 --open
```

Résultat

```
etienne@pop-os:~$ sudo nmap 10.0.0.0/24 -p 22 --open
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-11 18:34 EST
Nmap scan report for 10.0.0.18
Host is up (0.00050s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: F0:2F:74:1A:C4:F3 (Unknown)

Nmap scan report for 10.0.0.33
Host is up (0.054s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: DC:E9:94:90:23:0D (Unknown)

Nmap scan report for pop-os (10.0.0.243)
Host is up (0.00014s latency).

PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 256 IP addresses (8 hosts up) scanned in 5.68 seconds
```

Données recueillies

IP courant	Intervalle network	IP machines
10.0.0.243	10.0.0.0/24	10.0.0.33
		10.0.0.243

L'adresse qui nous intéresse ici est le 10.0.0.33 , car c'est sur cette adresse que nous effectuerons une attaque de type brute force.

Attaque brute force

Pour effectuer une attaque brute force, nous devons d'abord construire deux fichiers. Un sera en charge de contenir une énumération de mots de passe et l'autre celui des noms d'utilisateurs. Elles seront ensuite combinées pour accomplir le brute force sur toutes les combinaisons possibles. Ainsi, un fichier plus gros aura pour effet de prendre plus de temps à brute force.

Note

Une liste de mot de passe assez garni serait d'utilisé Rockyou.txt. Cependant, avoir recours à cette liste viendrait ajouter un temps considérable au test. Ainsi, nous allons nous limiter à quelques. options.

users.txt

Description

Contiens une liste de nom d'utilisateur potentiel.

Contenu

wow
root
toor
sudo
who
etienne
gandalf

Description

Contiens une liste de mot de passe potentiel.

Contenu

```
po
allo
123
allopo
bella
```

Pour effectuer, une attaque brute force, nous devons exécuter:

```
sudo nmap 10.0.0.33 -p 22 --script ssh-brute --script-args userdb=users.txt,passdb=passwords.txt
```

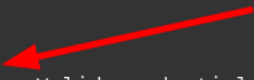
Résultat

```
etienne@pop-os:~/Udes/AUT2021/INF808$ sudo nmap 10.0.0.33 -p 22 --script ssh-brute --script-args userdb=users.txt,passdb=passwords.txt
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-11 18:58 EST
NSE: [ssh-brute] Trying username/password pair: wow:wow
NSE: [ssh-brute] Trying username/password pair: root:root
NSE: [ssh-brute] Trying username/password pair: toor:toor
NSE: [ssh-brute] Trying username/password pair: sudo:sudo
NSE: [ssh-brute] Trying username/password pair: who:who
NSE: [ssh-brute] Trying username/password pair: etienne:etienne
NSE: [ssh-brute] Trying username/password pair: gandalf:gandalf
NSE: [ssh-brute] Trying username/password pair: wow:po
NSE: [ssh-brute] Trying username/password pair: root:po
NSE: [ssh-brute] Trying username/password pair: toor:po
NSE: [ssh-brute] Trying username/password pair: sudo:po
NSE: [ssh-brute] Trying username/password pair: who:po
NSE: [ssh-brute] Trying username/password pair: etienne:po
NSE: [ssh-brute] Trying username/password pair: gandalf:po
NSE: [ssh-brute] Trying username/password pair: wow:allo
NSE: [ssh-brute] Trying username/password pair: root:allo
NSE: [ssh-brute] Trying username/password pair: toor:allo
NSE: [ssh-brute] Trying username/password pair: sudo:allo
NSE: [ssh-brute] Trying username/password pair: who:allo
NSE: [ssh-brute] Trying username/password pair: etienne:allo
NSE: [ssh-brute] Trying username/password pair: gandalf:allo
NSE: [ssh-brute] Trying username/password pair: wow:123
NSE: [ssh-brute] Trying username/password pair: root:123
NSE: [ssh-brute] Trying username/password pair: toor:123
NSE: [ssh-brute] Trying username/password pair: sudo:123
NSE: [ssh-brute] Trying username/password pair: who:123
NSE: [ssh-brute] Trying username/password pair: etienne:123
NSE: [ssh-brute] Trying username/password pair: gandalf:123
NSE: [ssh-brute] Trying username/password pair: wow:allopo
NSE: [ssh-brute] Trying username/password pair: root:allopo
NSE: [ssh-brute] Trying username/password pair: toor:allopo
NSE: [ssh-brute] Trying username/password pair: sudo:allopo
NSE: [ssh-brute] Trying username/password pair: who:allopo
NSE: [ssh-brute] Trying username/password pair: etienne:allopo
NSE: [ssh-brute] Trying username/password pair: gandalf:allopo
NSE: [ssh-brute] Trying username/password pair: wow:bella
NSE: [ssh-brute] Trying username/password pair: root:bella
NSE: [ssh-brute] Trying username/password pair: toor:bella
NSE: [ssh-brute] Trying username/password pair: sudo:bella
NSE: [ssh-brute] Trying username/password pair: who:bella
NSE: [ssh-brute] Trying username/password pair: etienne:bella
NSE: [ssh-brute] Trying username/password pair: gandalf:bella
Nmap scan report for 10.0.0.33
Host is up (0.054s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-brute:
|   Accounts:
|     gandalf:bella - Valid credentials
|_ Statistics: Performed 42 guesses in 11 seconds, average tps: 3.8
MAC Address: DC:E9:94:90:23:0D (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 12.17 seconds
```

```
PORT    STATE SERVICE
22/tcp  open  ssh
| ssh-brute:
|   Accounts:
|     gandalf:bella - Valid credentials
|_ Statistics: Performed 42 guesses in 18 seconds, average tps: 2.3
MAC Address: DC:E9:94:90:23:0D (Unknown)
```



La photo précédente illustre que nous avons bel et bien trouvé une correspondance entre le nom d'utilisateur **gandalf** et le mot de passe **bella**. D'ailleurs, il est possible d'effectuer une attaque brute force d'une autre façon. En effet, il est possible d'utiliser hydra qui se voit être plus rapide.

Pour utiliser hydra, exécuter:

```
sudo hydra -L users.txt -P passwords.txt ssh://10.0.0.33 -t 8
```

Résultat

```
etienne@pop-os:~/Udes/AUT2021/INF808$ sudo hydra -L users.txt -P passwords.txt ssh://10.0.0.33 -t 8
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or
for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-12-11 21:48:07
[DATA] max 8 tasks per 1 server, overall 8 tasks, 35 login tries (l:7/p:5), ~5 tries per task
[DATA] attacking ssh://10.0.0.33:22/
[22][ssh] host: 10.0.0.33  login: gandalf  password: bella
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-12-11 21:48:22
```

```
[22][ssh] host: 10.0.0.33  login: gandalf  password: bella
1 of 1 target successfully completed, 1 valid password found
```

Dans cette situation, nous conservons la même conclusion. C'est à dire, que le nom d'utilisateur **gandalf** et le mot de passe **bella** concordent ensemble.

Nous venons d'avoir la confirmation qu'avoir recours à l'authentification par utilisateur plutôt que par clef ssh est hautement risqué. Certes, les mots de passe étaient courts dans l'exemple. Toutefois, nous avons opté pour ce choix dans un contexte de démonstration. Un mot de passe plus compliqué peut éventuellement être brute forcé dépendamment du temps, des ressources et de l'intérêt que dispose l'attaquant.

Le modèle de sécurité que nous avons présenté permet de contrer ce type d'attaque en bloquant toutes demandes de connexions sans clef ssh.

Conséquences de l'utilisation du brute force après notre modèle appliqué

```
PORT    STATE SERVICE
22/tcp  open  ssh
|_ssh-brute: Password authentication not allowed
MAC Address: DC:E9:94:90:23:0D (Unknown)
```

Le brute force n'en reste pas moins impossible. Toutefois, il est vraiment plus difficile dans tirer profit dans un contexte où le serveur accepte uniquement les connexions par clef ssh en plus d'un passphrase. En effet, brute force une clef rsa de 4096 bits n'est pas impossible, mais il devient extrêmement non trivial de réussir un tel exploit. D'autant plus que dans un contexte de home serveur, il n'est pas **commun** de voir ce genre de tentative.

Pour aller plus loin

- ☐ Mettre en place une attaque dos
- ☐ Créer un script regroupant plus de configurations mentionnées afin d'automatiser la sécurisation du serveur
- ☐ Tester SSHGuard, Fail2ban and DenyHosts pour contrer les attaques par force brute

Références

Désactivé la connexion root SSH

Qu'est-ce qu'une clé SSH

Schéma d'authentification SSH

<https://www.reddit.com/r/linux4noobs/comments/9wlqsh/>

https://www.reddit.com/r/admincraft/comments/dw2lq9/how_safe_is_port_forwarding_a_server/

<https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server>

<https://serverfault.com/questions/475468/where-does-ufw-uncomplicated-firewall-save-command-line-rules-to>

<https://serverfault.com/questions/850659/securing-linux-servers-iptables-vs-fail2ban>

<https://security.stackexchange.com/questions/127092/hardening-linux-desktop-machine-against-people-from-my-household>

<https://security.stackexchange.com/questions/25137/what-are-some-steps-to-take-for-securing-a-linux-server-that-arent-in-this-list>

<https://serverfault.com/questions/258396/ultra-secure-linux-server-ssh-only/>

<https://acloudguru.com/blog/engineering/when-you-should-and-should-not-disable-root-login#:~:text=By creating a user with,the security of your system.>

<https://www.howtogeek.com/443156/the-best-ways-to-secure-your-ssh-server/>

<https://www.thegeekdiary.com/how-to-allow-ssh-with-empty-passwords-in-linux/#:~:text=PermitEmptyPasswords When password authentication is,accounts with empty password strings.&text=But if we set the,login with an empty password.>

https://www.reddit.com/r/PleX/comments/4lwu9l/server_being_claimed_by_someone_else/d3qrfat/

<https://linuxize.com/post/how-to-setup-a-firewall-with-ufw-on-ubuntu-18-04/>

<https://linuxize.com/post/how-to-set-up-ssh-keys-on-ubuntu-20-04/>

<https://askubuntu.com/questions/929934/how-can-i-create-multiple-ssh-keys>

<https://askubuntu.com/questions/362280/enter-ssh-passphrase-once>

<https://askubuntu.com/questions/174981/how-do-i-configure-ufw-to-allow-ssh-on-another-port>

<https://security.stackexchange.com/questions/141857/securing-ssh-connections>

<https://opensource.com/article/18/9/linux-iptables-firewalld>

<https://serverfault.com/questions/774569/how-can-allow-an-ssh-connection-only-over-vpn>

<https://dev.to/natterstefan/docker-tip-how-to-get-host-s-ip-address-inside-a-docker-container-5anh#:~:text=On Docker for Linux%2C the,you are using default networking.>