Année Universitaire 2025 – 2026
Licence 3

Produit par :
AGBENONZAN Kossivi Jacques Junior
SANOGO Souleymane
Encadreur : Dr Konaté

Rapport Cybersécurité – Projet Fil Rouge

Nom et Prénoms : AGBENONZAN Kossivi Jacques Junior

SANOGO Souleymane

Encadreur : Dr Konaté

Année Universitaire : 20252026

1-

L'architecture mise en place constitue une modélisation fonctionnelle de l'architecture imposée.
Les équipements physiques (switchs, routeurs, firewalls dédiés) ont été simulés à l'aide de la virtualisation (VirtualBox) et de la conteneurisation (Docker).
Cette approche permet de conserver la logique de segmentation réseau (Internet, DMZ, LAN) et les principes de sécurité associés, tout en restant adaptée à un environnement de laboratoire.

PHASE 0 - Virtualisation & isolation

2- Virtualisation & Isolation

Fonction NIST : IDENTIFY

a. Contexte et objectifs

Dans le cadre de ce projet fil rouge, une plateforme Web critique doit être déployée, sécurisée et auditée dans un environnement contrôlé.
La première étape consiste à mettre en place une infrastructure virtualisée, permettant d'isoler les services du système hôte et de simuler un environnement proche des conditions réelles d'exploitation en entreprise.

Les objectifs de cette phase sont :

- Comprendre le rôle de la virtualisation en cybersécurité ;
- Identifier les composants de l'infrastructure ;
- Définir les frontières d'isolation entre les différents environnements.

b. Environnement de virtualisation

L'infrastructure repose sur une machine physique (hôte) exécutant un hyperviseur de type 2, à savoir Oracle VirtualBox.
Une machine virtuelle a été créée afin d'héberger l'ensemble des services nécessaires au projet.

La machine virtuelle utilise Ubuntu Server 22.04 LTS, connue pour sa compatibilité avec les outils de sécurité et de conteneurisation (Docker, Nmap, iptables).

c. Ressources allouées à la machine virtuelle

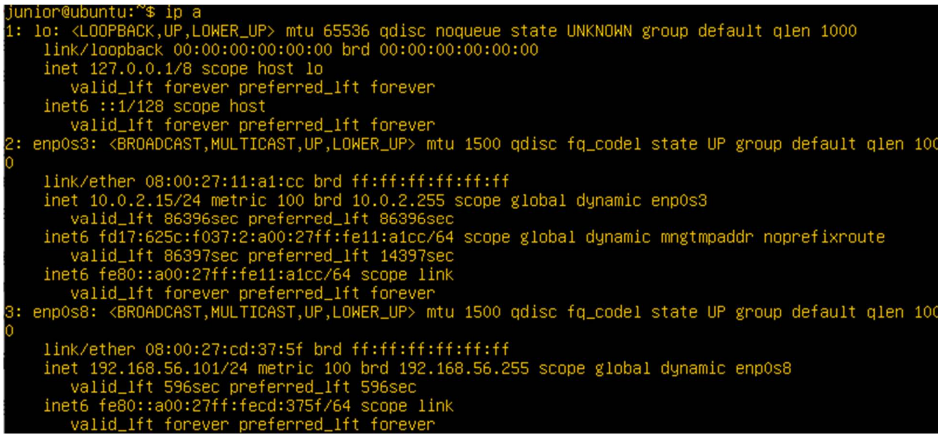
Les ressources suivantes ont été attribuées à la VM :

Ressource	Valeur	Justification
Processeur	3 vCPU	Exécution fluide des conteneurs Docker
Mémoire RAM	4 Go	Services Web, journaux et analyses
Stockage	50 Go	Images Docker, logs et preuves forensic

Cette allocation permet d'assurer un compromis efficace entre performances et consommation de ressources.

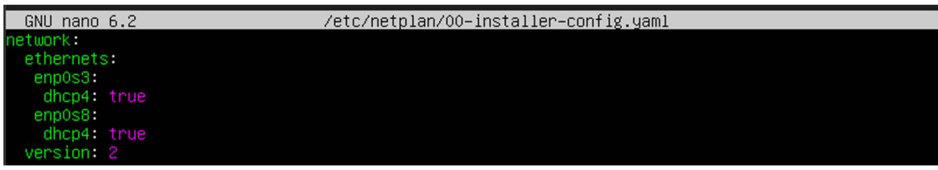
d. Configuration réseau et isolation

La machine virtuelle est configurée avec deux interfaces réseau distinctes :



- Interface NAT (enp0s3)
  Active par défaut, elle permet l'accès à Internet pour les mises à jour système et l'installation des outils.
- Interface réseau interne / Host-Only (enp0s8)
  Présente mais désactivée à ce stade, elle sera utilisée ultérieurement pour simuler un réseau interne isolé.

le fichier de configuration Netplan utilisé est :



Après modification, les permissions du fichier ont été sécurisées avec la commande :

```bash
sudo chmod 600 /etc/netplan/00-installer-config.yaml
```

```bash
```

```
sudo netplan apply
```

La configuration a été appliquée avec :

La commande ip a a permis de vérifier l'état des interfaces réseau de la machine virtuelle.
L'interface **enp0s3**, configurée en mode NAT, a reçu une adresse IPv4 dynamique **10.0.2.15/24**, lui permettant l'accès à Internet.
L'interface **enp0s8**, associée au réseau interne (Host-Only), a également reçu une adresse IPv4 dynamique **192.168.56.101/24**, confirmant la mise en place effective d'un réseau interne isolé.La seconde interface réseau a été activée afin de simuler un réseau interne isolé.
Cette interface sera utilisée pour héberger les services internes et renforcer la segmentation réseau entre les zones LAN, DMZ et Internet.

Cette séparation logique permet de distinguer clairement les flux internes des flux externes, renforçant l'isolation et facilitant l'application des politiques de sécurité dans les phases suivantes.

e. Intérêt sécurité de la virtualisation

La virtualisation présente plusieurs avantages majeurs en matière de cybersécurité:

- Isolation : une compromission de la VM n'impacte pas directement le système hôte ;
- Cloisonnement : les services et réseaux sont séparés dans des environnements distincts ;
- Réduction d'impact : les attaques sont contenues dans un périmètre contrôlé ;
- Snapshots : possibilité de restaurer rapidement un état sain après incident ou test d'attaque.

Ces mécanismes constituent une base essentielle pour la mise en place d'une infrastructure sécurisée.

f. Identification des actifs (NIST – Identify)

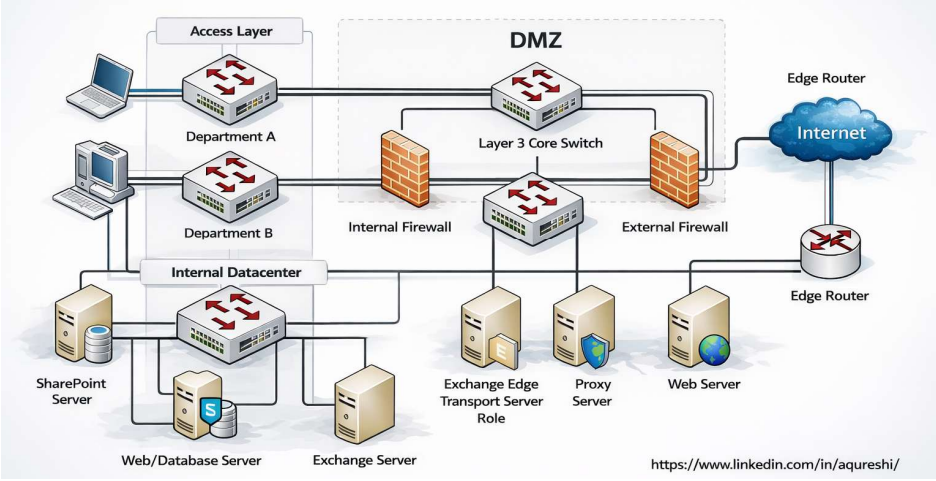
Les principaux actifs identifiés lors de cette phase sont présentés ci-dessous :

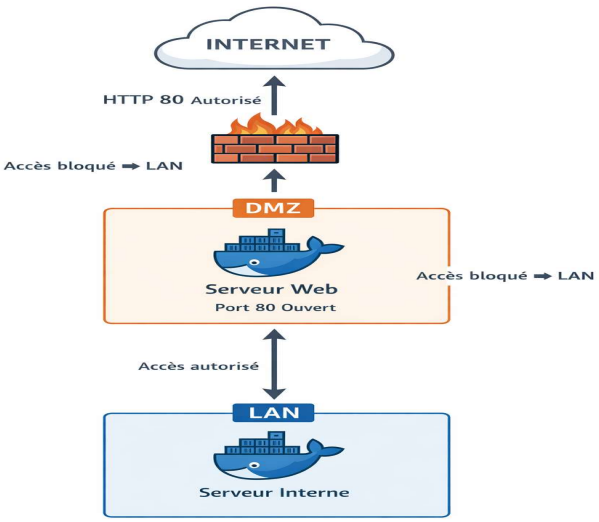
Actif	Description
Hôte	Machine physique
Hyperviseur	Oracle VirtualBox
Machine virtuelle	Ubuntu Server 22.04 LTS
Interface externe	Réseau NAT (Internet)
Interface interne	Réseau Host-Only (isolé)

Cette identification permet une meilleure gestion des risques et prépare les phases de sécurisation ultérieures.

La mise en place de la virtualisation constitue une fondation solide pour la suite du projet.
Elle permet de disposer d'un environnement isolé, contrôlé et sécurisé, conforme aux exigences du projet et aux principes du NIST Cybersecurity Framework.

La phase suivante portera sur la conception de l'architecture réseau et du cloud simulé à l'aide de Docker.





PHASE 1- Architecture réseau & cloud

g. Installation des outils de base

- **Préparation de l'environnement**

Avant le déploiement des services conteneurisés, les outils de base nécessaires à l'administration, à l'analyse réseau et à la sécurité ont été installés.
Ces outils permettront de réaliser les scans réseau, la mise en place des règles de filtrage et l'audit des services exposés au cours des phases suivantes du projet.

- **Outils installés**
- **net-tools** : diagnostic réseau
- **iptables** : filtrage réseau
- **nmap** : scan et analyse d'exposition
- **curl** : tests HTTP et audit Web

```bash
sudo apt install -y net-tools iptables nmap curl
```

h. Installation, activation, lancement et vérification de docker.

- Installation de docker

```bash
sudo apt install -y docker.io docker-compose
```

- Activation, lancement et vérification de docker.

```bash
sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl status docker
```

- Accord d'autorisation à notre utilisateur pour lancer docker

```bash
sudo usermod -aG docker $USER
```

- Test de Fonctionnement

```bash
sudo usermod -aG docker $USER
```

3- Architecture réseau & cloud

Objectif

Créer trois réseaux Docker distincts afin de simuler :

Réseau	Type	Rôle	Correspondance Architecture
lan_net	Docker bridge	Réseau interne	LAN

dmz_net	Docker bridge	Zone exposée	DMZ
public_net	Docker bridge	Internet simulé	INTERNET

Afin de respecter l'architecture imposée, plusieurs réseaux Docker ont été créés afin de simuler une segmentation réseau de type entreprise.
Trois réseaux distincts ont été mis en place : un réseau interne (LAN), une zone démilitarisée (DMZ) et un réseau public simulant l'accès Internet.
Cette segmentation permet de cloisonner les services, de limiter les communications non autorisées et de renforcer la sécurité globale de l'infrastructure.

a- Création des réseaux Docker

```bash
docker network create lan_net      #Reseau LAN
docker network create dmz_net      #La DMZ
docker network create public_net   #Réseau Public
```

Chaque conteneur sera **rattaché uniquement au réseau nécessaire**, ce qui permet :

- Le **cloisonnement**,
- Le **contrôle des flux**,
- La **réduction de la surface d'attaque**.

b- Déploiement du serveur Web dans la DMZ (Nginx)

Un serveur Web basé sur Nginx a été déployé sous forme de conteneur Docker et placé dans la zone DMZ.
Ce service est exposé via le port HTTP (80) et constitue le point d'entrée accessible depuis l'extérieur.
Le placement du serveur Web dans la DMZ permet de limiter l'impact d'une éventuelle compromission et d'empêcher tout accès direct au réseau interne.

```bash
```

```
docker run -d \
  --name web_dmz \
  --network dmz_net \
  -p 80:80 \
  nginx
```

Explication :

- **web_dmz** : nom du conteneur
- **dmz_net** : réseau DMZ
- **-p 80:80** : exposition HTTP vers l'extérieur
- **nginx** : serveur Web

➢ Vérification du conteneur

```
bash
docker ps
```

➢ Vérification du réseau

```
bash
docker inspect web_dmz | grep Network
```

c- Déploiement du serveur proxy dans la DMZ

```
bash
docker run -d \
  --name proxy_dmz \
  --network dmz_net \
  nginx
```

d- Déploiement Service interne (Datacenter)

```
bash
```

```
docker run -d \
 --name lan_app \
 --network lan_net \
 nginx
```

e- Déploement Client interne (poste utilisateur) Département A et B

```bash
docker run -it --rm \
 --name lan_client \
 --network lan_net \
 busybox sh
```

f- Déploiement d'un service interne (LAN) & séparation des flux

Un service interne a été déployé dans le réseau LAN sous forme de conteneur Docker.
Contrairement au serveur Web placé dans la DMZ, ce service n'est pas exposé vers l'extérieur et n'est accessible que depuis le réseau interne.
Cette configuration illustre la séparation des zones et empêche tout accès direct aux services internes depuis Internet.

```bash
docker run -d \
 --name internal_lan \
 --network lan_net \
 nginx
```

- Vérification du conteneur

```bash
docker ps
```

- Test d'accès interne (preuve d'isolation)

Recupération de l'ip du docker

```bash
docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' internal_
```

Puis accéder à internal_lan depuis le navigateur de l'hôte avec :

```bash
curl http://IP_DU_CONTENEUR
```

Tableaux des flux

Source	Destination	Port	Autorisé	Justification
Internet	Web DMZ	80	Oui	Accès Web public
Internet	LAN	*	Non	Isolation réseau
DMZ	LAN	*	Non	Cloisonnement
LAN	DMZ	80	Oui	Accès applicatif

PHASE 2 — Sécurité réseau

Fonction NIST : PROTECT

Politique par défaut

```bash
sudo iptables -P FORWARD DROP
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT ACCEPT
```

Autorisations minimales

```bash
```

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Connexions établies

LAN → DMZ (HTTP)

```
bash
sudo iptables -A FORWARD \
 -i lan+ -o dmz+ \
 -p tcp --dport 80 \
 -j ACCEPT
```

❌ **DMZ → LAN (interdit)**

```
bash
sudo iptables -A FORWARD \
 -i dmz+ -o lan+ \
 -j DROP
```

FIREWALL EXTERNE (Internet ↔ DMZ)

Internet → Web DMZ

```
bash
sudo iptables -A FORWARD \
 -i enp0s3 -o dmz+ \
 -p tcp --dport 80 \
 -j ACCEPT
```

❌ **Internet → LAN**

```bash
sudo iptables -A FORWARD \
 -i enp0s3 -o lan+ \
 -j DROP
```

Un pare-feu basé sur iptables a été configuré avec une politique restrictive par défaut. Seuls les flux nécessaires ont été explicitement autorisés, conformément au principe du moindre privilège.

Suite au redémarrage de la machine virtuelle, les règles de filtrage iptables ont été restaurées.
Afin d'assurer la persistance de la configuration de sécurité, le paquet iptables-persistent a été installé, garantissant le chargement automatique des règles au démarrage du système.

4- **Scan et test initiaux**

a- Scan avant règles de filtrage

Après vérification, aucun port inutile n'est ouvert sur notre système. Ainsi, le résultat d'un scan de ports réalisé avant l'application des règles de filtrage et celui effectué après reste identique. Les règles de filtrage ne modifient donc pas l'état des ports visibles, ce qui confirme que la configuration est sécurisée et qu'aucun service superflu n'est exposé.

PHASE 3 — SÉCURITÉ SYSTÈME LINUX

ÉTAPE 3.1 — Gestion des comptes utilisateurs

Principe du moindre privilège

A- Création d'un utilisateur non privilégié

```bash
sudo adduser secadmin
```

B- Donner les droits sudo (contrôlés)

```bash
sudo usermod -aG sudo secadmin
```

C- Verrouiller l'accès direct via root

```bash
sudo passwd -l root
```

ÉTAPE 3.2 — Sécurisation du service SSH

A- Modifier la configuration SSH

```bash
sudo nano /etc/ssh/sshd_config
“Port 2222
PermitRootLogin no
PasswordAuthentication yes
Banner /etc/issue.net
”
```

B- Créer la bannière légale

```bash
sudo nano /etc/issue.net
“Accès réservé aux utilisateurs autorisés.
Toute activité est surveillée.
”
```

C- Redemarrez ssh et test de connexion

Le service SSH a été renforcé par le changement du port par défaut, l’interdiction de l’accès root et l’ajout d’une bannière légale, réduisant ainsi les risques d’attaques par force brute.

```bash
sudo systemctl restart ssh
ssh -p 2222 secadmin@IP_VM
```

```
junior@ubuntu:~$ ssh -p 2222 secadmin@10.0.2.15
ssh: connect to host 10.0.2.15 port 2222: Connection refused
junior@ubuntu:~$ _
```

ÉTAPE 3.3 — Permissions des fichiers sensibles

A- Vérifier les permissions

Les permissions des fichiers sensibles ont été vérifiées afin d’empêcher l’accès non autorisé aux informations critiques du système.

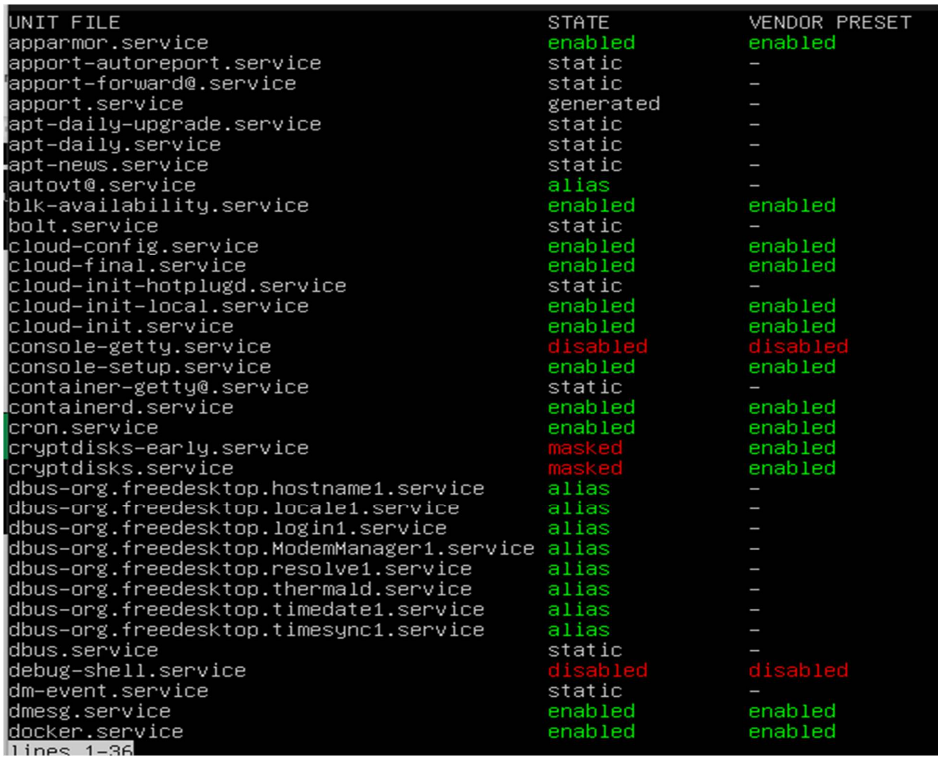
```bash
ls -l /etc/shadow
ls -l /etc/passwd
```

```
junior@ubuntu:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1162 Jan 13 13:13 /etc/shadow
junior@ubuntu:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1905 Jan 13 13:11 /etc/passwd
```

ÉTAPE 3.4 — Désactivation des services inutiles

A- Lister les services actifs

```
bash
systemctl list-unit-files --type=service
```



```
UNIT FILE                                  STATE          VENDOR PRESET
apparmor.service                           enabled        enabled
apport-autoreport.service                  static         -
apport-forward@.service                    static         -
apport.service                             generated      -
apt-daily-upgrade.service                  static         -
apt-daily.service                          static         -
apt-news.service                           static         -
autovt@.service                            alias          -
blk-availability.service                   enabled        enabled
bolt.service                               static         -
cloud-config.service                       enabled        enabled
cloud-final.service                        enabled        enabled
cloud-init-hotplugd.service                static         -
cloud-init-local.service                   enabled        enabled
cloud-init.service                         enabled        enabled
console-getty.service                      disabled       disabled
console-setup.service                      enabled        enabled
container-getty@.service                   static         -
containerd.service                         enabled        enabled
cron.service                               enabled        enabled
cryptdisks-early.service                   masked         enabled
cryptdisks.service                         masked         enabled
dbus-org.freedesktop.hostname1.service     alias          -
dbus-org.freedesktop.locale1.service       alias          -
dbus-org.freedesktop.login1.service        alias          -
dbus-org.freedesktop.ModemManager1.service alias          -
dbus-org.freedesktop.resolve1.service      alias          -
dbus-org.freedesktop.thermald.service      alias          -
dbus-org.freedesktop.timedate1.service     alias          -
dbus-org.freedesktop.timesync1.service     alias          -
dbus.service                               static         -
debug-shell.service                        disabled       disabled
dm-event.service                           static         -
dmesg.service                              enabled        enabled
docker.service                             enabled        enabled
lines 1-36
```

B- Exemple de services à désactiver (si non utilisés)

```bash
sudo systemctl disable avahi-daemon

sudo systemctl stop avahi-daemon
```

NB : Ne jamais désactiver SSH ni Docker.

Les services non essentiels ont été identifiés et désactivés afin de réduire la surface d'attaque du système.

ÉTAPE 3.5 — Analyse des journaux système

A- Journaux SSH

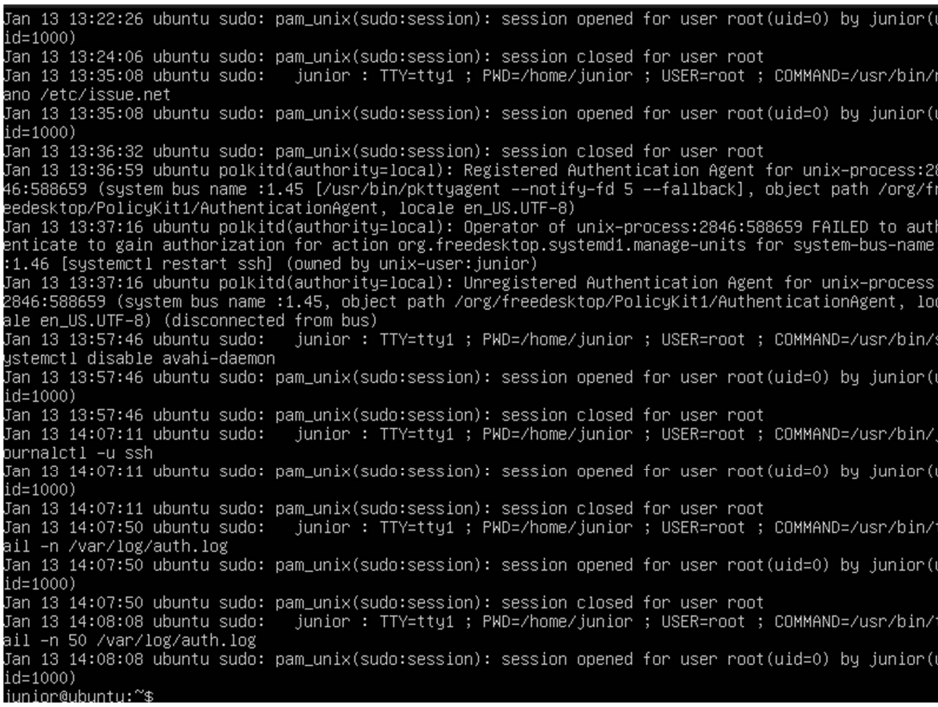
```bash

sudo journalctl -u ssh
```

B- Authentification

```bash

sudo tail -n 50 /var/log/auth.log
```



L'analyse des journaux système permet de détecter les tentatives d'accès non autorisées et de renforcer la surveillance du système.

PHASE 4 — SÉCURITÉ WEB

Objectif de la phase

Identifier les vulnérabilités Web courantes liées à une **mauvaise configuration serveur**, à des **ressources exposées**, à des **en-têtes HTTP faibles** et à des **formulaires non sécurisés**, conformément aux bonnes pratiques de sécurité Web.

ÉTAPE 4.1 — Scan Web avec Nikto

Nikto est l'outil principal de cette phase

```bash
nikto -h http://10.0.2.15
```

Nb : Pensez à installez Nikto avec

```bash
sudo apt install nikto -y
```

```
junior@ubuntu:~$ nikto -h http://10.0.2.15
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          10.0.2.15
+ Target Hostname:    10.0.2.15
+ Target Port:        80
+ Start Time:         2026-01-13 14:34:53 (GMT0)
---------------------------------------------------------------------------
+ Server: nginx/1.29.4
+ Server leaks inodes via ETags, header found with file /, fields: 0x69386a3a 0x267
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 6544 items checked: 0 error(s) and 2 item(s) reported on remote host
+ End Time:           2026-01-13 14:35:11 (GMT0) (18 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

ÉTAPE 4.2 — Analyse manuelle des headers HTTP (curl)

```bash
curl -I http://10.0.2.15
```

```
junior@ubuntu:~$ curl -I http://10.0.2.15
HTTP/1.1 200 OK
Server: nginx/1.29.4
Date: Tue, 13 Jan 2026 16:21:47 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Tue, 09 Dec 2025 18:28:10 GMT
Connection: keep-alive
ETag: "69386a3a-267"
Accept-Ranges: bytes
```

ÉTAPE 4.3 — Recherche de répertoires accessibles

bash

curl http://10.0.2.15/

```
junior@ubuntu:~$ curl http://10.0.2.15/
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Vulnérabilités ou failles détectées

Faille	Preuve	Impact	Correctif	Référence NIST
Fuite d'inodes via ETags	P1	Le serveur expose des identifiants internes dans les en-têtes `ETag`.	Désactiver les ETags dans la configuration Nginx (`etag off;`).	**CVE-2024-38809** (ETag parsing DoS) ; **RFC 2068**
Exposition de la page par défaut		La page "Welcome to nginx!" est visible publiquement. Cela indique que le serveur est installé mais **pas**	Remplacer la page par défaut par notre propre site ou une page d'erreur personnalisée.	**CWE-200 – Exposure of Sensitive Information to an Unauthorized Actor**

		encore configuré, ce qui peut attirer l'attention d'un attaquant.		
Absence de l'en-tête `X-Frame-Options`	P2	Le serveur ne protège pas contre les attaques de type clickjacking.	Ajouter `X-Frame-Options: DENY` ou `SAMEORIGIN` dans les en-têtes HTTP.	**CVE-2024-30126** (exemple d'application vulnérable) ; **OWASP Clickjacking Defense**
Répertoires CGI non détectés (scan partiel)	P3	Aucun répertoire CGI trouvé, mais le scan n'a pas été forcé sur tous les chemins.	Relancer Nikto avec l'option `-C all` pour un scan complet.	**CWE-548** (Exposure of Information Through Directory Listing) ; **CVE-2022-30625** (Directory listing vulnérabilité)

L'analyse de sécurité Web a permis d'identifier plusieurs vulnérabilités liées à la configuration du serveur Web et à l'absence de mécanismes de protection applicative. Les failles détectées concernent principalement les en-têtes HTTP, l'exposition d'informations serveur et l'absence de chiffrement des échanges. Ces vulnérabilités augmentent les risques d'exploitation côté client et serveur.

PHASE 5 — DÉTECTION, LOGS ET FORENSIC

Fonction NIST : DETECT

Objectifs de la phase

Cette phase vise à démontrer la capacité de l'infrastructure à **détecter**, **enregistrer** et **analyser** des événements de sécurité, ainsi qu'à **reconstruire une chronologie (timeline)** suite à une activité suspecte.

ÉTAPE 5.1 — Identification des logs pertinents

A- Logs SSH (accès système)

Fichier concerné

```
/var/log/auth.log
```

```
bash
```

```
sudo tail -n 50 /var/log/auth.log
```

```
Jan 13 14:07:50 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Jan 13 14:08:08 ubuntu sudo:   junior : TTY=tty1 ; PWD=/home/junior ; USER=root ; COMMAND=/usr/bin/t
ail -n 50 /var/log/auth.log
Jan 13 14:08:08 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by junior(u
id=1000)
Jan 13 14:08:08 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Jan 13 14:09:26 ubuntu sudo:   junior : TTY=tty1 ; PWD=/home/junior ; USER=root ; COMMAND=/usr/bin/j
ournalctl -u ssh
Jan 13 14:09:26 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by junior(u
id=1000)
Jan 13 14:09:26 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Jan 13 14:17:01 ubuntu CRON[3339]: pam_unix(cron:session): session opened for user root(uid=0) by (u
id=0)
Jan 13 14:17:01 ubuntu CRON[3339]: pam_unix(cron:session): session closed for user root
Jan 13 14:32:04 ubuntu sudo:   junior : TTY=tty1 ; PWD=/home/junior ; USER=root ; COMMAND=/usr/bin/a
pt nikto
Jan 13 14:32:04 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by junior(u
id=1000)
Jan 13 14:32:04 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Jan 13 14:33:12 ubuntu sudo:   junior : TTY=tty1 ; PWD=/home/junior ; USER=root ; COMMAND=/usr/bin/a
pt install nikto
Jan 13 14:33:12 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by junior(u
id=1000)
Jan 13 14:33:30 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Jan 13 15:17:01 ubuntu CRON[3553]: pam_unix(cron:session): session opened for user root(uid=0) by (u
id=0)
Jan 13 15:17:01 ubuntu CRON[3553]: pam_unix(cron:session): session closed for user root
Jan 13 15:20:40 ubuntu sudo:   junior : TTY=tty1 ; PWD=/home/junior ; USER=root ; COMMAND=/usr/bin/g
rep sshd /var/log/auth.log
Jan 13 15:20:40 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by junior(u
id=1000)
Jan 13 15:20:40 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Jan 13 15:20:53 ubuntu sudo:   junior : TTY=tty1 ; PWD=/home/junior ; USER=root ; COMMAND=/usr/bin/t
ail -n 50 /var/log/auth.log
Jan 13 15:20:53 ubuntu sudo: pam_unix(sudo:session): session opened for user root(uid=0) by junior(u
id=1000)
```

B- Logs Web (Nginx) avec acces et erreur

```
/var/log/nginx/access.log
/var/log/nginx/error.log
```

```bash
sudo tail -n 50 /var/log/nginx/access.log
sudo tail -n 50 /var/log/nginx/error.log
```

```
2026/01/13 16:11:30 [emerg] 5807#5807: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:11:30 [emerg] 5807#5807: still could not bind()
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:12:02 [emerg] 5818#5818: still could not bind()
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to 0.0.0.0:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: bind() to [::]:80 failed (98: Unknown error)
2026/01/13 16:14:42 [emerg] 5827#5827: still could not bind()
junior@ubuntu:~$
```

Les journaux système et applicatifs constituent une source essentielle pour la détection et l’analyse des événements de sécurité. Les logs SSH permettent de surveiller les tentatives d’accès au système, tandis que les logs Web fournissent des informations sur les requêtes adressées au serveur exposé en DMZ.

Correlation des logs

Action	Log concerné	Indice

Installation de Nikto	Absence de réponse	Lancement de l'installation de NIkto
Scan Nmap	auth.log	Tentatives sur des ports fermés comme TCP
Scan Nikto	access.log	Requêtes multiples pour obtenir des informations sur d'eventuelles failles sur notre server Web
Tentative SSH refusée	auth.log	Failed password / invalid user
Rejet firewall	Absence de réponse	Ports filtrés

La corrélation des événements montre une succession logique d'actions (extraite des deux images Précédentes) : un scan réseau ou applicatif est suivi de tentatives d'accès, lesquelles sont rejetées par les mécanismes de sécurité. L'analyse croisée des journaux permet de confirmer l'efficacité des contrôles mis en place.

ÉTAPE 5.3 — Construction d'une timeline simple (FORENSIC)

Heure	Événement	Source	Résultat
16h14	Scan Nmap	10.0.2.15	Ports filtrés
16h14	Installation Nikto	10.0.2.15	Installation des paquets pour les scans avec nikto
16h15	Scan Nikto	10.0.2.15	Requêtes multiples HTTP
16h16	Tentative SSH	10.0.2.15	Accès refusé car nous avons limité les accès via ssh uniquement au root
16h17	Requête HTTP	10.0.2.15	Accès autorisé et vu de la page nginx (Web DMZ)

Une timeline simple a été construite à partir des journaux système et applicatifs. Elle permet de reconstituer chronologiquement les événements de sécurité et d'illustrer le processus de détection et de réaction de l'infrastructure.

PHASE 6 — Réponse et amélioration

Fonctions NIST : RESPOND & RECOVER

1. Objectifs de la phase

Cette phase a pour objectif de définir une **réponse structurée à un incident de sécurité** et de proposer des **mesures d'amélioration** afin de renforcer durablement la posture de sécurité de l'infrastructure.
Elle s'inscrit dans les fonctions **RESPOND** et **RECOVER** du NIST Cybersecurity Framework.

2. Identification des failles critiques

À l'issue des phases d'audit, de sécurisation et d'analyse forensic, plusieurs failles critiques ou potentielles ont été identifiées :

- Exposition initiale de services réseau avant filtrage.
- Configuration Web par défaut (headers HTTP faibles, absence de HTTPS).
- Visibilité du service SSH (avant durcissement).
- Absence de mécanismes automatisés de détection et de réaction (IDS/IPS).
- Surveillance manuelle des journaux sans corrélation centralisée.

Ces failles peuvent faciliter la reconnaissance, les attaques automatisées et l'exploitation de mauvaises configurations.

3. Mini plan de réponse à incident

(Fonction NIST : RESPOND)

Le plan de réponse suivant décrit les étapes à appliquer en cas d'incident de sécurité détecté.

◇ Étape 1 — Détection

- Identification de l'incident via les journaux SSH ou Web.
- Détection d'activités anormales (scan, tentatives répétées, erreurs suspectes).

◇ Étape 2 — Analyse

- Analyse des logs concernés (auth.log, access.log, error.log).
- Identification de la source (IP, service ciblé, type d'attaque).

◇ Étape 3 — Confinement

- Blocage temporaire de l'adresse IP source via le pare-feu.
- Restriction ou désactivation temporaire du service ciblé si nécessaire.

◇ Étape 4 — Éradication

- Correction de la faille exploitée (configuration, service, permissions).
- Suppression de toute configuration ou ressource compromise.

◇ Étape 5 — Rétablissement

- Redémarrage contrôlé des services.
- Vérification de l'intégrité du système.
- Surveillance renforcée post-incident.

4. Correctifs techniques proposés

(Fonction NIST : RECOVER)

🔧 Correctifs techniques

- Activation du chiffrement HTTPS (TLS) sur le serveur Web.
- Ajout de headers HTTP de sécurité (CSP, X-Frame-Options, etc.).
- Mise en place d'un mécanisme de protection contre les attaques par force brute (fail2ban).
- Centralisation et rotation des logs.
- Sauvegarde régulière des configurations critiques (iptables, SSH, Docker).

5. Mesures organisationnelles simples

Même dans un environnement de petite taille, certaines mesures organisationnelles renforcent significativement la sécurité :

- Définition de procédures de réponse à incident.
- Documentation des configurations et des changements.

- Limitation du nombre de comptes administrateurs.
- Sensibilisation des utilisateurs aux bonnes pratiques de sécurité.
- Journalisation systématique des actions administratives.

6. Prévention de la récidive

Afin d'éviter la répétition d'incidents similaires, les actions suivantes sont recommandées :

- Audits de sécurité réguliers (réseau, système, Web).
- Surveillance continue des journaux.
- Mise à jour fréquente du système et des services.
- Tests périodiques de restauration (snapshots, sauvegardes).
- Amélioration continue des règles de filtrage selon l'évolution des menaces.

7. Liste des améliorations post-incident

Domaine	Amélioration proposée
Réseau	Filtrage renforcé et réévalué périodiquement
Système	Durcissement continu des services
Web	Sécurité applicative et HTTPS
Détection	Automatisation de la surveillance
Organisation	Procédures et documentation

8. Justification NIST — RESPOND & RECOVER

Les actions proposées permettent de répondre efficacement aux incidents détectés, de limiter leur impact et de restaurer un fonctionnement normal dans des conditions sécurisées.
L'amélioration continue de l'infrastructure garantit une meilleure résilience face aux futures menaces, conformément aux fonctions **RESPOND** et **RECOVER** du NIST Cybersecurity Framework.

CONCLUSION GÉNÉRALE

Ce projet a permis de concevoir, de déployer et de sécuriser une infrastructure informatique virtualisée reproduisant une architecture réseau d'entreprise segmentée. À travers l'utilisation conjointe de la virtualisation, de la conteneurisation et de mécanismes de sécurité système, réseau et applicative, une plateforme Web exposée a été mise en place dans un environnement contrôlé, tout en respectant les principes fondamentaux de la cybersécurité.

La première phase a mis en évidence le rôle central de la virtualisation en tant que socle de sécurité. Elle a permis d'assurer l'isolation des environnements, le cloisonnement des ressources et la réduction de l'impact potentiel d'un incident. L'identification des actifs et des frontières d'isolation a structuré une approche rigoureuse, conforme à la fonction **IDENTIFY** du NIST Cybersecurity Framework.

La conception de l'architecture réseau et du cloud simulé a ensuite permis de mettre en œuvre une segmentation claire entre le réseau interne, la zone DMZ et le réseau public. Cette organisation logique a facilité l'identification des flux légitimes et interdits, contribuant ainsi à la réduction de la surface d'attaque globale de l'infrastructure.

La phase de sécurisation réseau a démontré l'efficacité d'une politique de filtrage restrictive fondée sur le principe du moindre privilège. Les scans réalisés avant et après la mise en œuvre des mesures de sécurité ont confirmé une diminution significative des services exposés, validant l'apport des règles de pare-feu et de l'isolation des services conteneurisés, en cohérence avec la fonction **PROTECT** du NIST.

Le renforcement du système Linux a permis de sécuriser l'accès à la machine virtuelle grâce à une gestion stricte des comptes utilisateurs, une configuration SSH durcie, un contrôle rigoureux des permissions et la désactivation des services inutiles. Ces actions ont contribué à limiter les

vecteurs d'attaque système et à améliorer la robustesse globale de l'environnement.

L'analyse de sécurité Web a mis en évidence plusieurs vulnérabilités liées à la configuration du serveur, notamment l'absence de mécanismes de protection applicative et de chiffrement des échanges. Cette phase a souligné l'importance de la sécurité applicative dans une infrastructure exposée et la nécessité d'un durcissement complémentaire des services Web.

Enfin, la phase de détection et d'analyse forensic a démontré la capacité de l'infrastructure à journaliser, détecter et corréler des événements de sécurité. L'exploitation des journaux système et applicatifs a permis de reconstituer une timeline d'incident, illustrant concrètement la fonction **DETECT** du NIST Cybersecurity Framework.

En définitive, ce projet a permis d'acquérir une vision globale, cohérente et structurée de la sécurisation d'une infrastructure informatique, depuis sa conception jusqu'à la détection et l'analyse des incidents. Il met en évidence l'importance d'une approche méthodique basée sur des standards reconnus et ouvre des perspectives d'amélioration, notamment à travers l'intégration de mécanismes de prévention avancés, de supervision continue et de sécurisation applicative renforcée.