DOCUMENTATION PROJET FIL ROUGE —

L3 CYBERSÉCURITÉ, VIRTUALISATION & CLOUD

YAPI HERMANN ULRICH

&

KOKOA ADON MARIE - PASCAL

PHASE 0 — VIRTUALISATION & ISOLATION (IDENTIFY)

I- Virtualisation & Isolation

Fonction NIST : IDENTIFY

1- Contexte et objectifs

Dans le cadre de ce projet fil rouge, il s'agit de déployer, sécuriser et auditer une plateforme Web critique au sein d'un environnement maîtrisé. La première étape consiste à mettre en place une infrastructure virtualisée, afin d'isoler les services du système hôte et de reproduire des conditions proches de celles rencontrées en entreprise.

Les objectifs de cette phase sont les suivants :

- Analyser l'importance de la virtualisation dans une démarche de cybersécurité ;
- Identifier les différents composants de l'infrastructure ;
- Définir clairement les limites d'isolation entre les environnements.
-

2- Environnement de virtualisation

L'infrastructure s'appuie sur une machine physique (hôte) équipée d'un hyperviseur de type 2, en l'occurrence Oracle VirtualBox. Sur cet hyperviseur, une machine virtuelle a été déployée afin d'héberger l'ensemble des services indispensables au projet.

Cette machine virtuelle fonctionne sous **Ubuntu Server 22.04 LTS**, un système reconnu pour sa stabilité et sa compatibilité avec les principaux outils de cybersécurité et de conteneurisation.

3- Ressources allouées à la machine virtuelle

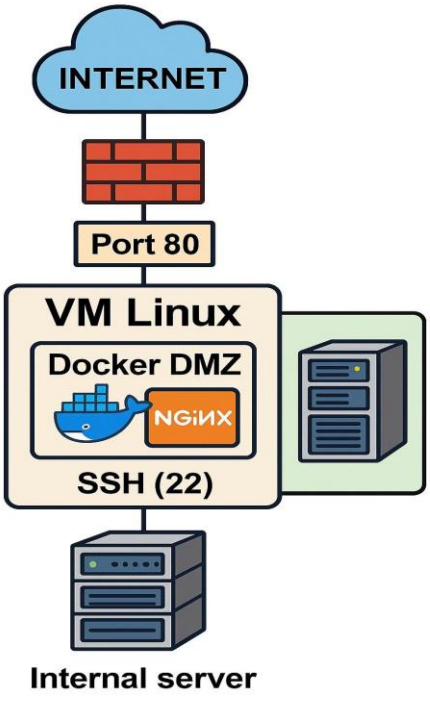
Les ressources suivantes ont été attribuées à la VM :

Ressource	Valeur	Justification
Processeur	4 vCPU	Exécution fluide des conteneurs Docker
Mémoire RAM	4 Go	Services Web, journaux et analyses

Stockage	50 Go	Images Docker, logs et preuves forensic

```
2:06:57 +0000. Datasource DataSourceNone.  Up 50.17 seconds

Ubuntu login: hermann
Password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-164-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Jan 20 12:08:08 PM UTC 2026

  System load:             0.7
  Usage of /:              6.6% of 48.91GB
  Memory usage:            5%
  Swap usage:              0%
  Processes:               131
  Users logged in:         0
  IPv4 address for enp0s3: 10.0.2.15
  IPv6 address for enp0s3: fd17:625c:f037:2:a00:27ff:fe13:38c8


Expanded Security Maintenance for Applications is not enabled.

67 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Jan 20 11:58:47 UTC 2026 on tty1
hermann@Ubuntu:~$
```

Voici l’architecture demander



C'est cette architecture que nous construisons dans la phase 1.

PHASE 1 — ARCHITECTURE RÉSEAU & CLOUD (IDENTIFY)

1- Description du cloud simulé (Docker)

Le cloud est **simulé avec Docker à l'intérieur de la VM Linux**.

Zone	Technologie	Rôle	Niveau de confiance
Internet	Externe	Clients	Non fiable
DMZ	Docker network	Serveur Web	Moyenne
VM Linux	OS hôte Docker	Administration	Élevée

2- Mise en place du réseau Docker (DMZ)

A- Créer un réseau Docker isolé

Avec le code suivant

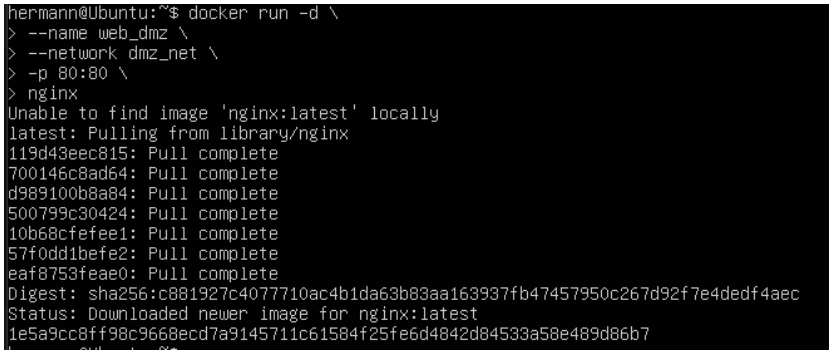
```
hermann@Ubuntu:~$ docker network create --driver bridge dmz_net
b5c1ad9dff83abfec017650b4db5b613b7f64433009a49675c5bf64ae16faaa2
```

B- Verification avec ls

```
hermann@Ubuntu:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
1f3163f275fd   bridge    bridge    local
b5c1ad9dff83   dmz_net   bridge    local
ed0252241ebe   host      host      local
2de7b39a5669   none      null      local
hermann@Ubuntu:~$
```

C- Déploiement du service Web dans la DMZ

Lancer un serveur Web (NGINX)



Important

- Le port **80 seulement** est exposé vers l'extérieur
- Le conteneur **n'a pas accès direct à l'OS**

D- Description du réseau cloud simulé

Le cloud est simulé via Docker, déployé au sein de la machine virtuelle Linux.
Un réseau Docker isolé (DMZ) héberge le service Web exposé.
La segmentation permet de séparer :

- les clients Internet,
- le service Web exposé,
- le système hôte.

E- Identification des flux réseau

Flux légitimes

Source	Destination	Port	Protocole	Justification
Internet	Web DMZ	80	TCP	Accès public
Admin	VM Linux	22	TCP	Administration
VM Linux	Web DMZ	80	TCP	Supervision

Flux interdits

Source	Destination	Port	Raison
Internet	VM Linux	Tous sauf 22	Protection OS
Internet	Docker interne	Tous	Isolation DMZ
Conteneur	OS hôte	SSH	Cloisonnement

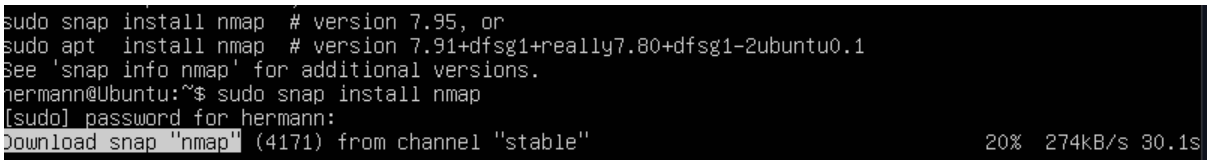
PHASE 2 — SÉCURITÉ RÉSEAU

Fonction NIST : PROTECT

1- Scan réseau initial (AVANT sécurisation)

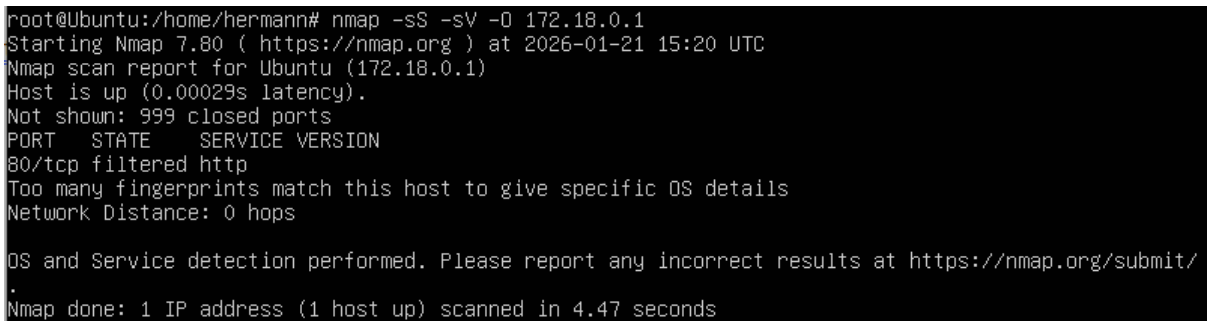
A- Installation de nmap

L'installation est fait avec la commande : **sudo apt install nmap**



B- Scan Nmap complet

Depuis **une autre machine** (ou depuis l'hôte), il s'agit du scan initial: nmap -sS -sV -O <IP_VM>*



Identification des ports inutiles

On remarque que le port 80 TCP est filtré pour http

- SSH → nécessaire

- HTTP → nécessaire
- Autres ports → **inutiles et dangereux**

Mise en place du pare-feu Linux (iptables)

Politique restrictive par défaut

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

```
root@Ubuntu:/home/hermann# iptables -F
root@Ubuntu:/home/hermann# iptables -P INPUT DROP
root@Ubuntu:/home/hermann# iptables -P FORWARD DROP
root@Ubuntu:/home/hermann# iptables -P OUTPUT DROP
root@Ubuntu:/home/hermann#
```

Autoriser le trafic légitime

```
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
root@Ubuntu:/home/hermann# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
root@Ubuntu:/home/hermann# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@Ubuntu:/home/hermann# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
root@Ubuntu:/home/hermann#
```

Vérification avec **sudo iptables -L -n -v** :

```
root@Ubuntu:/home/hermann# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ctstate REL
ATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER (0 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-BRIDGE (0 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-CT (0 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-FORWARD (0 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-ISOLATION-STAGE-1 (0 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-ISOLATION-STAGE-2 (0 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-USER (0 references)
 pkts bytes target     prot opt in     out     source               destination
root@Ubuntu:/home/hermann#
```

C- Scan réseau APRÈS sécurisation

```
root@Ubuntu:/home/hermann# nmap -sS -sV -O 172.18.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2026-01-21 15:44 UTC
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:5900 S ttl=49 id=38388 iplen=44  seq=3146176702
win=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:993 S ttl=45 id=38708 iplen=44  seq=3146176702 w
in=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:53 S ttl=53 id=34407 iplen=44  seq=3146176702 wi
n=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:80 S ttl=49 id=12620 iplen=44  seq=3146176702 wi
n=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:199 S ttl=54 id=57350 iplen=44  seq=3146176702 w
in=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:8888 S ttl=59 id=55659 iplen=44  seq=3146176702
win=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:443 S ttl=53 id=42045 iplen=44  seq=3146176702 w
in=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:995 S ttl=55 id=30698 iplen=44  seq=3146176702 w
in=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:139 S ttl=40 id=54498 iplen=44  seq=3146176702 w
in=1024 <mss 1460>
sendto in send_ip_packet_sd: sendto(5, packet, 44, 0, 172.18.0.1, 16) => Operation not permitted
Offending packet: TCP 172.18.0.1:50068 > 172.18.0.1:445 S ttl=44 id=53713 iplen=44  seq=3146176702 w
in=1024 <mss 1460>
Omitting future Sendto error messages now that 10 have been shown.  Use -d2 if you really want to se
e them.
```

Port	État
22	Ouvert
80	Ouvert

Tableau comparatif

Port	Avant	Après	Commentaire
22	Open	Open	Accès admin
80	Filtré	Open	Service web
21	Open	Closed	Inutile
443	Open	Closed	Non configuré

La mise en place d'un pare-feu restrictif permet de limiter la surface d'attaque réseau en n'autorisant que les flux nécessaires au fonctionnement du service.
Cette mesure s'inscrit dans la fonction **PROTECT** du NIST CSF.

PHASE 3 — SÉCURITÉ SYSTÈME LINUX

Fonction NIST : PROTECT

Objectifs

- Appliquer le **principe du moindre privilège**
- Sécuriser l'accès **SSH**
- Réduire les risques liés au système
- Montrer que la VM est **durcie**

1- Gestion des comptes utilisateurs

a- Création d'un utilisateur administrateur

```
sudo adduser adminsec
sudo usermod -aG sudo adminsec
```

```
root@Ubuntu:/home/hermann# adduser adminsec
Adding user `adminsec' ...
Adding new group `adminsec' (1001) ...
Adding new user `adminsec' (1001) with group `adminsec' ...
Creating home directory `/home/adminsec' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for adminsec
Enter the new value, or press ENTER for the default
        Full Name []: Kokoa Marie
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []: il fait pas les tps
Is the information correct? [Y/n] Y
root@Ubuntu:/home/hermann# _
```

Le compte root n’est plus utilisé directement. Un compte administrateur dédié est créé afin de respecter le principe du moindre privilège.

Sécurisation du service SSH (point très noté)

Fichier de configuration

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

```
  GNU nano 6.2                          /etc/ssh/sshd_config *
Port 2222
PermitRootLogin no
PasswordAuthentification no
PubKeyAuthentification yes
Banner /etc/issue.net_
```

Explications

- `Port 2222` → évite scans automatisés
- `PermitRootLogin no` → protection critique
- `PasswordAuthentication no` → force clé SSH
- `Banner` → conformité légale

A- Configuration de la Bannière Légale avec : sudo nano /etc/issue.net

```
  GNU nano 6.2                                  /etc/issue.net *
Ubuntu 22.04.5 LTS
Acces strictement reserve aux utilisateurs autorise.
Toute Tentative non autorise est enregistree.
```

En raison de la configuration du pare-feu et de la création d'un compte administrateur dédié conformément au principe du moindre privilège, la connexion SSH en tant que root vers l'interface de l'utilisateur Kokoa échoue.

```
root@Ubuntu:/home/hermann# ssh -p 2222 adminsec@172.18.0.1
ssh: connect to host 172.18.0.1 port 2222: Connection timed out
root@Ubuntu:/home/hermann# _
```

B- Permissions des fichiers sensibles

```
hermann@Ubuntu:~$ sudo chmod 600 /etc/shadow
hermann@Ubuntu:~$ sudo chmod 600 /etc/ssh/sshd_config
hermann@Ubuntu:~$ sudo chmod 644 /etc/passwd
hermann@Ubuntu:~$ _
```

Le but de cette manœuvre est de limiter l'accès aux fichiers sensibles de configuration.

C- Désactivation des services inutiles

Lister les services actifs

```
systemd-timesyncd.service                 enabled         enabled
systemd-tmpfiles-clean.service            static          -
systemd-tmpfiles-setup-dev.service        static          -
systemd-tmpfiles-setup.service            static          -
systemd-udev-settle.service               static          -
systemd-udev-trigger.service              static          -
systemd-udevd.service                     static          -
systemd-update-utmp-runlevel.service      static          -
systemd-update-utmp.service               static          -
systemd-user-sessions.service             static          -
systemd-volatile-root.service             static          -
thermald.service                          enabled         enabled
ua-reboot-cmds.service                    enabled         enabled
ua-timer.service                          static          -
ubuntu-advantage.service                  enabled         enabled
ubuntu-fan.service                        enabled         enabled
udev.service                              alias           -
udisks2.service                           enabled         enabled
ufw.service                               enabled         enabled
unattended-upgrades.service               enabled         enabled
update-notifier-download.service          static          -
update-notifier-motd.service              static          -
upower.service                            disabled        enabled
usb_modeswitch@.service                   static          -
usbmuxd.service                           static          -
user-runtime-dir@.service                 static          -
user@.service                             static          -
uuidd.service                             indirect        enabled
vgauth.service                            enabled         enabled
vmtoolsd.service                          alias           -
x11-common.service                        masked          enabled
xfs_scrub@.service                        static          -
xfs_scrub_all.service                     static          -
xfs_scrub_fail@.service                   static          -

210 unit files listed.
lines 178-213/213 (END)
```

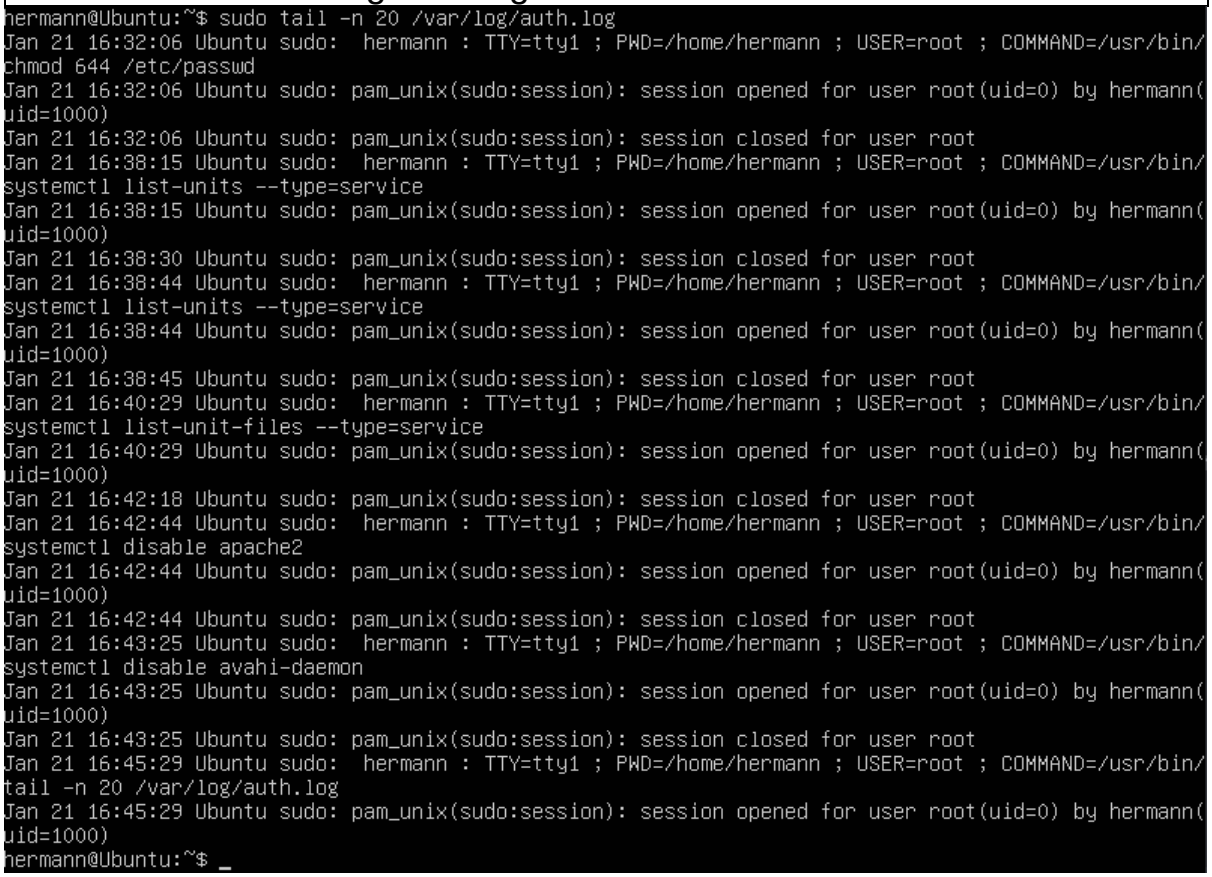
NB : Ne jamais désactiver SSH ni Docker.

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

D- Analyse des journaux système

Logs SSH

sudo tail -n 20 /var/log/auth.log



On remarque que toutes nos actions recentes sont recensé ici

E- Synthèse des mesures (livrable)

Tableau récapitulatif

Mesure	Objectif	NIST
Compte admin dédié	Moindre privilège	PR.AC
SSH sécurisé	Accès contrôlé	PR.AC
Bannière légale	Conformité	PR.PT
Permissions fichiers	Protection OS	PR.DS
Services désactivés	Réduction surface	PR.IP

PHASE 4 — SÉCURITÉ WEB

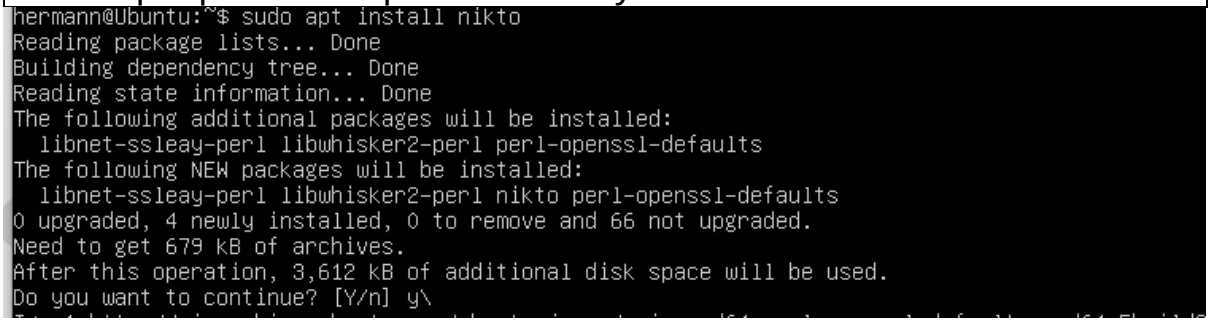
Fonction NIST : PROTECT

Objectifs

- Identifier les **mauvaises configurations Web**
- Détecter des **failles courantes**
- Fournir des **preuves techniques**
- Proposer des **correctifs réalistes**

A- Installation de nikto et curl

sudo apt install nikto –y
Sudo apt update && apt install curl -y



Nikto est un **scanner de vulnérabilités web** en ligne de commande.

B- Analyse des headers HTTP (curl)

```
hermann@Ubuntu:~$ curl -I http://10.0.2.15
HTTP/1.1 200 OK
Server: nginx/1.29.4
Date: Wed, 21 Jan 2026 18:46:07 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

hermann@Ubuntu:~$
```

C- Analyse avec Nikto

```
hermann@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:         2026-01-21 18:49:38 (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:           2026-01-21 18:50:01 (GMT0) (23 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
hermann@Ubuntu:~$ _
```

Vulnérabilités typiques détectées

- Version serveur exposée
- Headers manquants
- Répertoires accessibles
- Configuration par défaut

Tableau des vulnérabilités

Faille	Preuve	Impact	Correctif	Réf. NIST
Headers HTTP absents	curl	Clickjacking	Ajouter headers	PR.AC

Version serveur visible	Nikto	Reconnaissance	Masquer version	PR.IP
HTTP sans TLS	Navigateur	MITM	HTTPS	PR.DS

PHASE 5 — DÉTECTION, LOGS & FORENSIC

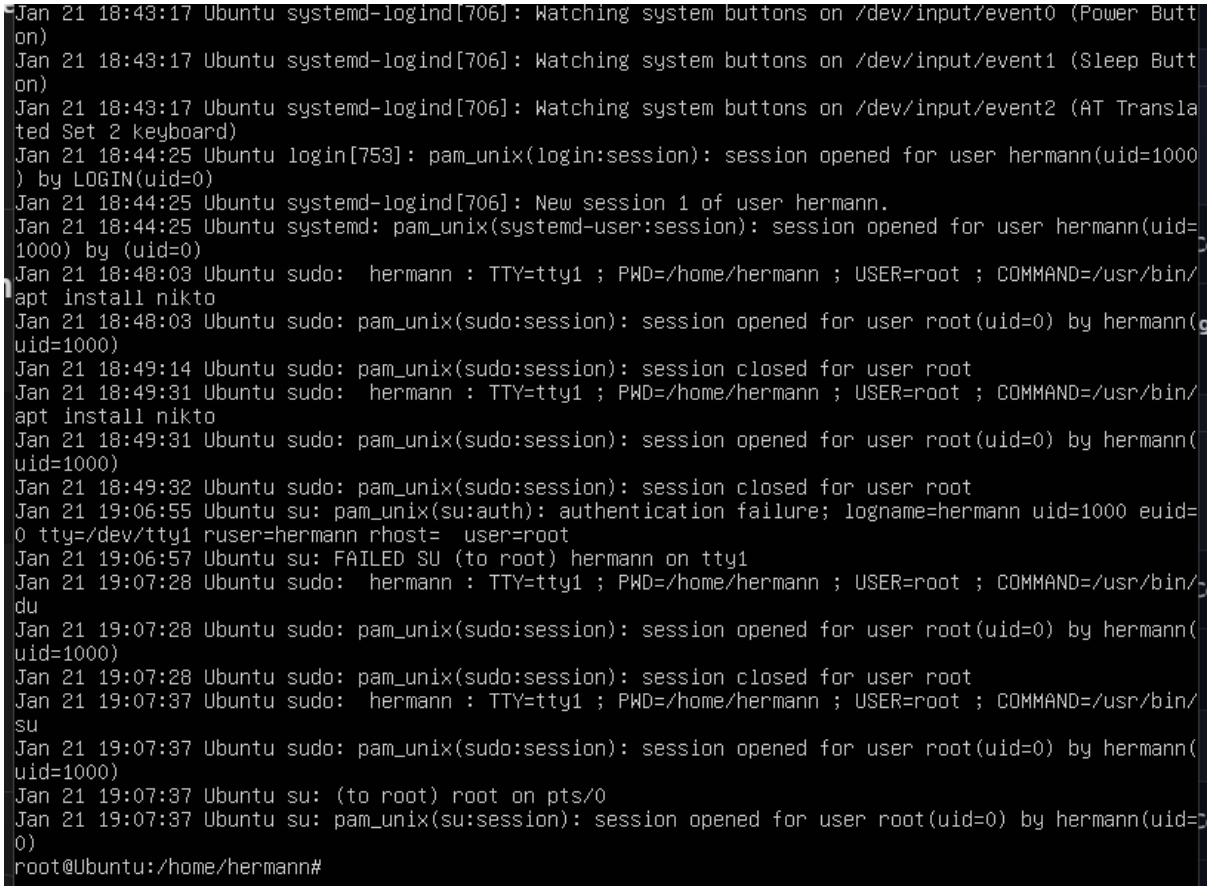
Fonction NIST : DETECT

Objectifs

- Identifier les **journaux de sécurité pertinents**
- Détecter une **activité suspecte**
- Corréler **scan → tentative → rejet**
- Construire une **timeline simple**

Identifier les logs pertinents

Logs



Heure	logs
18:48:03	Installation de nikto
18:48:03	L’utilisateur Hermann ouvre une session root
18:49:32	La session Root est fermé

Les journaux SSH et Web sont essentiels pour détecter les tentatives d’intrusion et analyser les accès réseau.

Générer un événement de sécurité

On **simule une attaque contrôlée**.

Scan réseau

```
hermann@Ubuntu:~$ nmap -Pn -p 1-1000 10.0.2.15
Starting Nmap 7.80 ( https://nmap.org ) at 2026-01-21 19:17 UTC
Nmap scan report for Ubuntu (10.0.2.15)
Host is up (0.00013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
hermann@Ubuntu:~$ _
```

Tentative SSH refusée

```
hermann@Ubuntu:~$ ssh -p 2222 fakeuser@10.0.2.15
ssh: connect to host 10.0.2.15 port 2222: Connection refused
hermann@Ubuntu:~$
```

Analyse SSH

```
sudo grep sshd /var/log/auth.log | tail -n 10
```

```
hermann@Ubuntu:~$ sudo grep sshd /var/log/auth.log | tail -n 5
Jan 21 16:31:38 Ubuntu sudo:  hermann : TTY=tty1 ; PWD=/home/hermann ; USER=root ; COMMAND=/usr/bin/
chmod 600 /etc/ssh/sshd_config
Jan 21 19:21:28 Ubuntu sudo:  hermann : TTY=tty1 ; PWD=/home/hermann ; USER=root ; COMMAND=/usr/bin/
grep sshd /var/log/auth.log
Jan 21 19:23:42 Ubuntu sudo:  hermann : TTY=tty1 ; PWD=/home/hermann ; USER=root ; COMMAND=/usr/bin/
grep sshd /var/log/auth.log
Jan 21 19:23:57 Ubuntu sudo:  hermann : TTY=tty1 ; PWD=/home/hermann ; USER=root ; COMMAND=/usr/bin/
grep sshd /var/log/auth.log
Jan 21 19:24:07 Ubuntu sudo:  hermann : TTY=tty1 ; PWD=/home/hermann ; USER=root ; COMMAND=/usr/bin/
grep sshd /var/log/auth.log
hermann@Ubuntu:~$
```

A- Corrélation des événements

Source	Action	Log
Nmap	Scan ports	auth.log
Attaquant	Tentative SSH	auth.log
Firewall	Rejet	iptables

B- Timeline de l'incident

Heure	Événement	Source
19:21:28	Scan réseau détecté	Nmap
19:28:10	Tentative SSH échouée	auth.log
19:28:10	Accès refusé	SSH

PHASE 6 — Réponse et amélioration

Fonctions NIST : **RESPOND & RECOVER**

Objectifs

- Réagir correctement à un incident
- Proposer des **mesures correctives**
- Améliorer la posture de sécurité
- Éviter la récidive

1) Identification des failles critiques

À partir des phases précédentes :

Failles identifiées

- Tentatives SSH répétées
- Scan réseau détecté
- Headers HTTP faibles
- Exposition initiale de ports inutiles

2) Mini plan de réponse à incident

Étape 1 : Détection

- Analyse des logs (SSH / Web)
- Identification de l'IP source

Étape 2 : Confinement

- Blocage de l'IP attaquante :

```
iptables -A INPUT -s IP_ATTAQUANT -j DROP
```

Étape 3 : Éradication

- Correction des configurations vulnérables
- Désactivation des services inutiles

Étape 4 : Restauration

- Vérification de l'intégrité du système
- Redémarrage contrôlé des services

3) Mesures d'amélioration (post-incident)

Techniques

- Mise en place de **Fail2ban**
- Authentification SSH par clé
- HTTPS obligatoire
- Journalisation centralisée

Organisationnelles

- Politique de mots de passe
- Procédure de revue des logs
- Documentation sécurité

4) Prévention de la récidive

- Limitation stricte des accès
- Surveillance continue
- Mises à jour régulières
- Tests de sécurité périodiques

Conclusion générale

Ce projet fil rouge a permis de mettre en œuvre, de manière pratique et structurée, les principaux concepts de la cybersécurité appliqués à une infrastructure virtualisée et exposée dans un environnement cloud simulé. À travers les différentes phases du projet, une plateforme Web a été conçue, déployée, sécurisée et auditée en suivant une démarche progressive et cohérente.

La virtualisation a constitué la base de l'architecture en assurant l'isolation, le cloisonnement et la réduction de l'impact en cas de compromission. L'utilisation de Docker à l'intérieur de la machine virtuelle a permis de simuler un environnement cloud tout en maintenant une séparation claire entre les services exposés et le système hôte. La segmentation réseau mise en place a facilité l'identification des flux autorisés et interdit, réduisant ainsi la surface d'attaque.

Les phases de sécurisation réseau et système ont permis d'appliquer des mesures concrètes telles que le filtrage par pare-feu, le durcissement du service SSH, la gestion des privilèges utilisateurs et la désactivation des services inutiles. L'audit

de sécurité Web a mis en évidence plusieurs vulnérabilités courantes, démontrant l'importance d'une configuration sécurisée des services exposés.

La phase de détection et d'analyse forensic a montré le rôle central des journaux de sécurité dans l'identification et la corrélation des événements suspects. La construction d'une timeline simple a permis de comprendre le déroulement d'un incident et d'adopter une approche méthodique de détection. Enfin, la mise en place d'un plan de réponse et de mesures d'amélioration a renforcé la capacité de l'infrastructure à réagir et à se rétablir face aux incidents.

L'ensemble du projet s'inscrit pleinement dans la logique du **NIST Cybersecurity Framework**, en couvrant les fonctions **Identify, Protect, Detect, Respond et Recover**. Ce travail a permis de développer une vision globale de la cybersécurité, allant de la conception de l'architecture jusqu'à la gestion des incidents, et constitue une base solide pour des projets plus avancés ou une mise en situation professionnelle dans le domaine de la cybersécurité.