# Sécurité

Initiation à la sécurité

Damien Mougenot

25/03/2025

Binôme : Frédéric Sturm

# Premier TP : T.P. Système

# T.P. Netcat

## Sommaire



## Introduction

Ce rapport détaille les travaux réalisés dans le cadre du TP sur l’utilisation de Netcat et la compréhension des connexions réseau et des attaques potentielles.

L'objectif principal était de mettre en place différentes connexions client-serveur et d'analyser des situations de sécurité telles que les chevaux de Troie.

# Compréhension des attentes

L'objectif principal de ce TP était de se familiariser avec Netcat et de comprendre son utilisation dans le contexte de la sécurité réseau. Nous avons dû réaliser plusieurs exercices de création de connexions entre une machine client et une machine serveur, d'attaque de type cheval de Troie, ainsi que l'analyse des connexions réseau à l'aide de Netstat.

Nous avons aussi analysé les résultats obtenus via Netcat et avons vérifié comment ces connexions pouvaient être exploitées dans un cadre de sécurité.

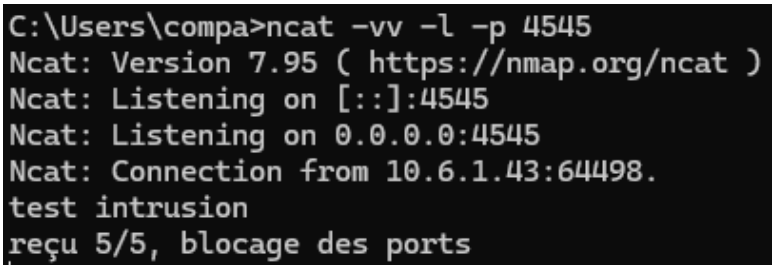
# Étapes de réalisation des objectifs

## Création d'une connexion Client - Serveur

```
C:\Users\compa>ncat -vv www.google.fr 80
Ncat: Version 7.95 ( https://nmap.org/ncat )
NCAT DEBUG: Using trusted CA certificates from C:\Program Files (x86)\Nmap\ca-bundle.crt.
libnsock nsock_iod_new2(): nsock_iod_new (IOD #1)
libnsock nsock_connect_tcp(): TCP connection requested to 172.217.18.195:80 (IOD #1) EID 8
libnsock nsock_trace_handler_callback(): Callback: CONNECT SUCCESS for EID 8 [172.217.18.195:80]
Ncat: Connected to 172.217.18.195:80.
libnsock nsock_iod_new2(): nsock_iod_new (IOD #2)
libnsock nsock_read(): Read request from IOD #1 [172.217.18.195:80] (timeout: -1ms) EID 18
libnsock nsock_readbytes(): Read request for 0 bytes from IOD #2 [peer unspecified] EID 26
```

Netcat a permis de se connecter avec succès au serveur **www.google.fr** sur le port **80**. Cette connexion établit une base pour envoyer des requêtes HTTP et observer les réponses du serveur, ce qui peut être utile pour comprendre le fonctionnement des protocoles réseau comme HTTP.

## Création d'un « Chat »





En mettant un serveur en écoute sur un port spécifique (4545) avec la commande **nc -vv -L -p 4545**, nous avons pu créer un chat simple. La connexion au serveur a été réalisée via un navigateur à l'adresse **http://<Adresse IP>:4545**. Cette phase nous a permis de voir en pratique comment un serveur pouvait accepter des connexions et les afficher dans Netcat.

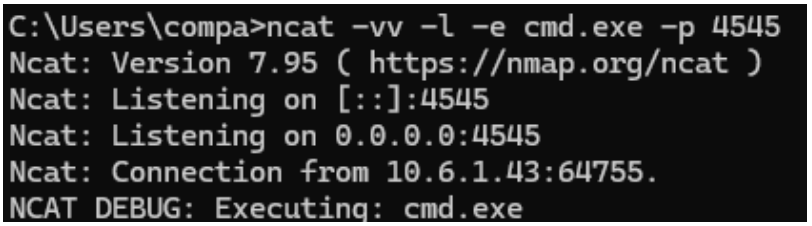
## Création d'un Cheval de Troie

Pour simuler une attaque de type cheval de Troie, nous avons mis la machine victime en écoute avec la commande :

```
nc -vv -L -e cmd.exe -p 4545
```

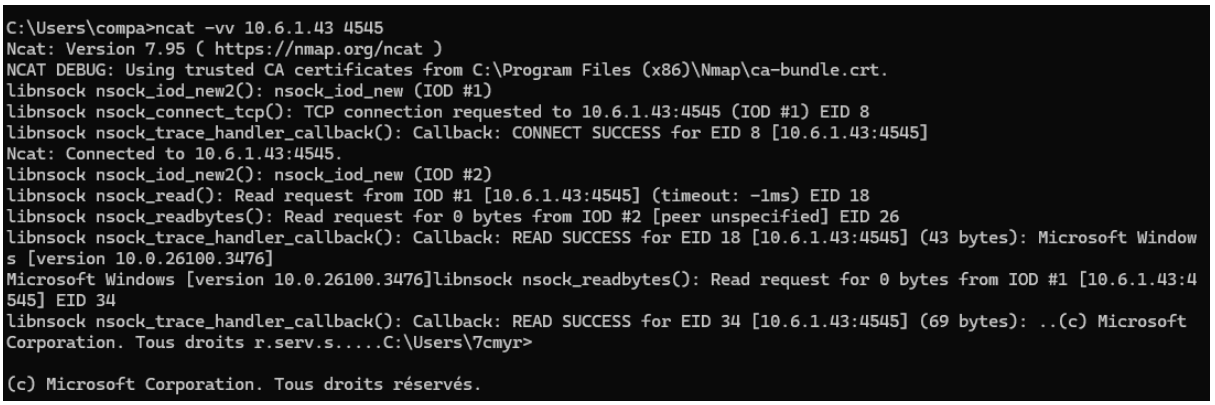
Le client (attaquant) s'est ensuite connecté à la victime via la commande :

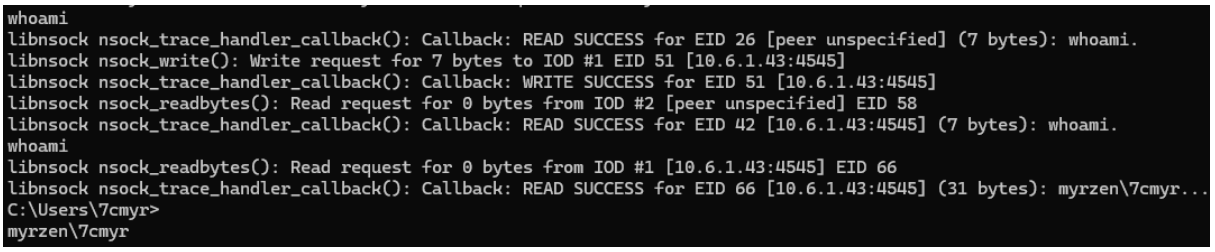
```
nc -vv <IP de la victime> 4545
```



Dans un premier temps j’ai joué le rôle de la victime, laissant alors ouvert mon port. Mon binôme a alors pu se connecter à mon ordinateur pour y effectuer des commandes. Dans le cadre actuel, nous avons juste décidé de faire chacun notre tour un whoami pour montrer que cela fonctionne bien.

> Dans un second temps j’ai joué le rôle de l’attaquant, me connectant à mon tour au pc de la victime. Cela a permis d'exécuter des commandes sur la machine victime à distance.

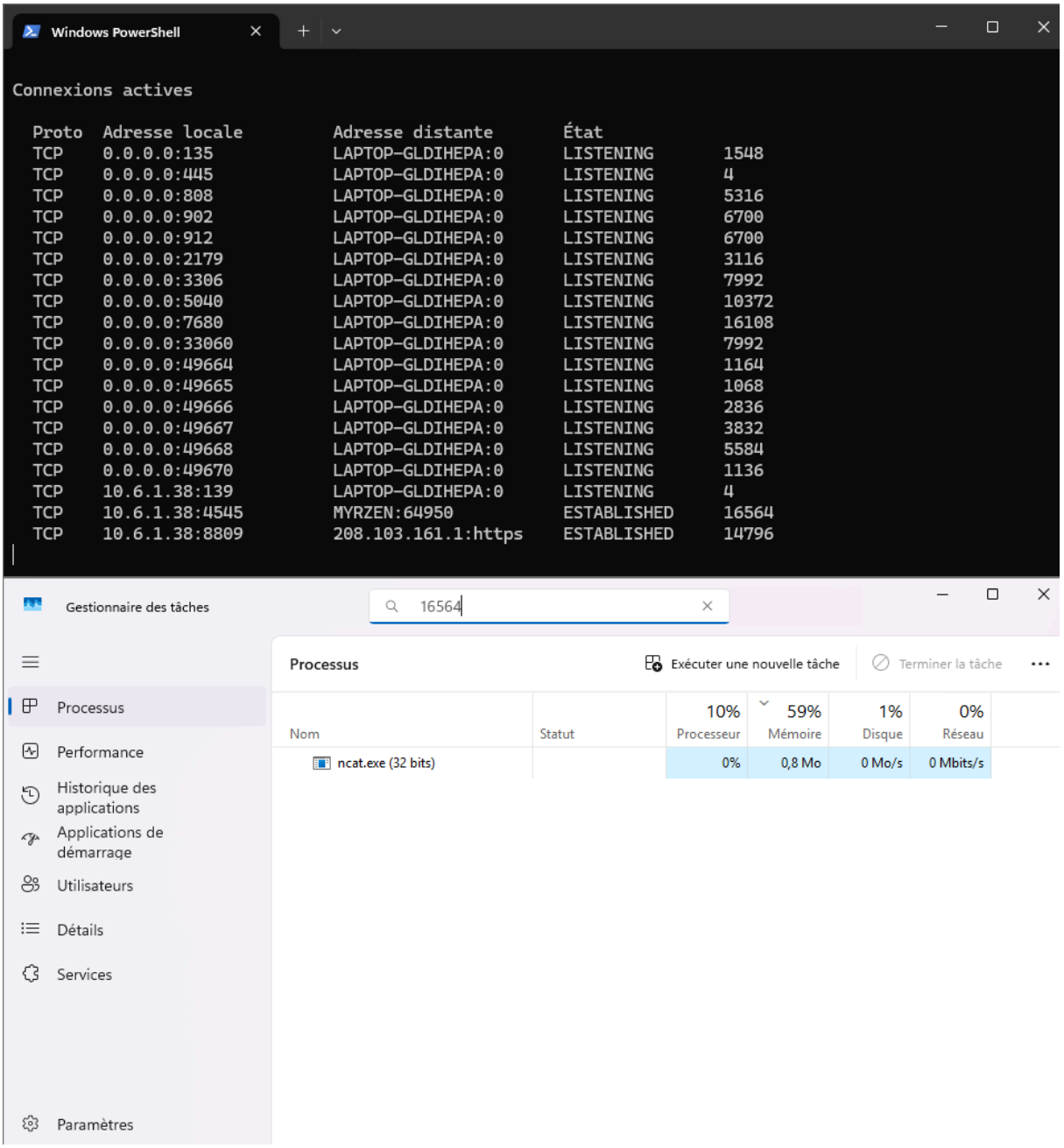



## Analyse avec Netstat

Nous avons utilisé Netstat pour observer les connexions réseau actives et identifier les processus associés aux connexions. La commande suivante a été utilisée sur Windows :

```
netstat -ao
```




Nous pouvons alors apercevoir qu'un certain "MYRZEN"(Binôme) est connecté à l'ordinateur sur le port 4545, ce qui n'est pas censé être normal si nous n'étions pas dans le cadre d'un exercice. Si l'on fait une recherche avec le PID 16564, on se rend compte que le processus ncat.exe est ouvert.

# Analyse des résultats obtenus et conclusion

## Résultats des connexions Client - Serveur

Les connexions HTTP ont été établies avec succès, et les résultats dans Netcat ont montré les détails de la connexion, comme les requêtes envoyées et les réponses du serveur.

### Création d'un « Chat »

Le serveur en écoute sur le port 4545 a fonctionné comme prévu, et a permis de tester la communication entre client et serveur via Netcat.

### Création d'un Cheval de Troie

La simulation d'un cheval de Troie a montré comment un attaquant pouvait prendre le contrôle d'un système à distance en exécutant des commandes sur la machine victime. La commande `cmd.exe` a été lancée avec succès sur la machine victime après la connexion de l'attaquant.

### Netstat et observation des processus

L'utilisation de Netstat a permis de visualiser les connexions réseau et les processus associés. Cela a renforcé la compréhension de la manière dont les connexions réseau sont liées à des programmes spécifiques.

### Conclusion

Ce TP a permis de comprendre l'importance des outils comme Netcat et Netstat dans l'analyse des connexions réseau et la détection des attaques potentielles. La simulation de connexions sécurisées et d'attaques a mis en évidence des risques réels pour la sécurité des réseaux, notamment avec l'utilisation de chevaux de Troie pour obtenir un accès distant. Les outils observés sont essentiels dans l'audit de la sécurité réseau.

# Deuxième TP : T.P. Système

## Sommaire

3. Étapes de réalisation
    - TP 2 : Hachage et Authentification
    - TP 3 : Chiffrement base64
    - TP 4 : Propriétés des fonctions de hachage
4. Analyse des résultats
5. Conclusion

## Compréhension des attentes

Ce TP visait à comprendre les principes fondamentaux de la cryptographie, notamment le hachage, l'authentification, la différence entre codage et chiffrement, ainsi que les propriétés spécifiques des fonctions de hash. Il s'agissait également de mettre en évidence les faiblesses de certains systèmes de sécurité mal implémentés, comme le stockage de mots de passe en hash.

## Méthodologie utilisée

- Recherche en ligne sur des bases de hachage inversées pour retrouver un mot de passe.
- Utilisation de l'outil PowerShell pour le calcul de hachage de fichiers.
- Création de fichiers textes simples pour les tests (avec bloc-notes).

## Etapes de réalisation

### T.P. : 2 / Hachage et Authentification (Suite)

Le mot de passe de l'utilisateur est badpassword.


On a pu le trouver car c'est un mot de passe fréquents et donc, par conséquent, déjà connu et répertorié dans des bases de données publiques, malgré le fait que

ce soit du SHA1.

## T.P. 3 : Chiffrement

```
PS C:\Users\compa> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("SWwgbmUgZmF1dCBwYXMgY29uZm
9uZHJlIENvZGFnZSBldCBDaGlmZnJlbWVudCAh"))
Il ne faut pas confondre Codage et Chiffrement !
```

Le message en clair est : “Il ne faut pas confondre Codage et Chiffrement !”

Nous avons également pu le déchiffrer facilement car la Base64 est du codage qui, contrairement au chiffrement, est bien plus simple à déchiffrer.

## T.P. 4 / Propriété des fonctions de Hash

Voici le premier Hash que j’obtiens après avoir écrit dans mon fichier : “Voici un texte sympatique”

```
PS C:\Users\compa> Get-FileHash C:\Users\compa\OneDrive\Bureau\Cours\B2\Sécurité/texte.txt -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          06D26573BECC4281DE65DA7DD6DA7406C9C36078FD424076D1213616DEA6A1A5       C:\Users\compa\OneDrive
```

Et voici ce que l’on obtient lorsque l’on modifie qu’un seul caractère donc : “voici un texte sympatique”

```
PS C:\Users\compa> Get-FileHash C:\Users\compa\OneDrive\Bureau\Cours\B2\Sécurité/texte.txt -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          8B6F552EB3C162EC1F814DB2FA5950FCBE768B7AC208EDB92A484CCA061B1815       C:\Users\compa\OneDrive
```

Le hash est désormais complètement différent même pour une minuscule modification. C’est ce qu’on appelle l’effet avalanche et cela permet de ne jamais pouvoir déchiffrer facilement.

Voici ce qu’il se produit lorsqu’on modifie le nom du fichier :

```
PS C:\Users\compa> Get-FileHash C:\Users\compa\OneDrive\Bureau\Cours\B2\Sécurité/test.txt -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          8B6F552EB3C162EC1F814DB2FA5950FCBE768B7AC208EDB92A484CCA061B1815       C:\Users\compa\OneDriv
```

Le Hash ne change pas car c’est le contenu du fichier qui est haché et non pas le fichier lui-même et son nom.

## Analyse des résultats

- Le **hachage** est fiable uniquement avec des mots de passe complexes et du **salage**.
- Le **Base64** n'est pas un moyen de protection, juste une transformation.
- Le **hachage réagit fortement** à la moindre modification du contenu, ce qui est souhaité pour détecter des altérations.

## Conclusion

Ce TP montre que :

- Un hash faible peut être cassé facilement si les mots de passe sont dans des dictionnaires publics.
- Le Base64 est insuffisant pour sécuriser un message.
- Les fonctions de hash sont sensibles à la moindre modification, ce qui les rend utiles pour l’intégrité des données.

Des outils simples comme PowerShell suffisent à tester et observer, augmentant notre compréhension des mécanismes de base de la sécurité informatique.

# Troisième TP : Administration SSH (étape 1 et 2)

## Sommaire

1. Compréhension des attentes
2. Méthodologie utilisée
3. Étapes de réalisation
    - Objectif 1 : Authentification par mot de passe
    - Objectif 2 : Authentification par clé publique/privée
4. Environnement et outils
5. Analyse et conclusion

# 1. Compréhension des attentes

Ce TP vise à mettre en place une connexion SSH sécurisée entre deux machines virtuelles Linux : une jouant le rôle de serveur et l'autre de client. Trois objectifs principaux sont demandés :

- Connexion SSH par mot de passe
- Désactivation de l'authentification par mot de passe
- Connexion via une paire de clés publique/privée

---

# 2. Méthodologie utilisée

La réalisation s'est faite par étapes progressives. Les tests ont été faits depuis le client vers le serveur, en adaptant la configuration SSH côté serveur.

- Utilisation du service `openssh-server` pour la connexion
- Installation du service `openssh-client` du côté du client
- Utilisation de la commande `ssh-copy-id` pour l'ajout de clé publique

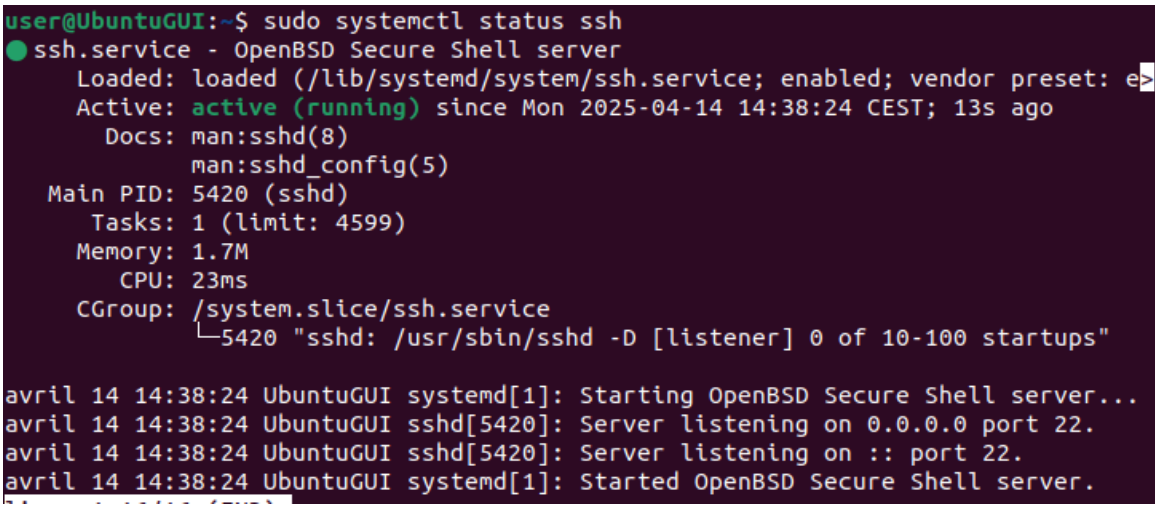
# 3. Étapes de réalisation

## Etape 1 et 2 : Connexion SSH par mot de passe

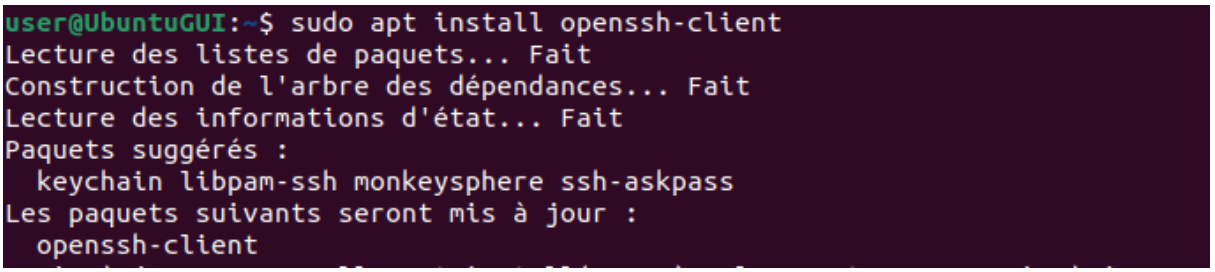
1. Installation de OpenSSH sur le serveur :

```
sudo apt update
sudo apt install openssh-server
```

2. Vérification du fonctionnement :




3. Installation d'openssh-client sur la machine virtuelle client




4. Connexion depuis le client après la désactivation de l'authentification par mot de passe :

```
user@UbuntuGUI:~$ ssh-copy-id user@192.168.56.102
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be established.
ED25519 key fingerprint is SHA256:maitJj7QxxyTRXSpgw1dyT/6/K8uQ9fMOhEVJS0qECY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
user@192.168.56.102's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'user@192.168.56.102'"
and check to make sure that only the key(s) you wanted were added.
```

Voici ce que l'on obtient lorsqu'on teste de se connecter. On ne nous demande plus de mot de passe.

```
user@UbuntuGUI:~$ ssh 'user@192.168.56.102'
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-33-generic x86_64)

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

La maintenance de sécurité étendue pour Applications n'est pas activée.

455 mises à jour peuvent être appliquées immédiatement.
324 de ces mises à jour sont des mises à jour de sécurité.
Pour afficher ces mises à jour supplémentaires, exécuter : apt list --upgradable

Activez ESM Apps pour recevoir des futures mises à jour de sécurité supplémenta
res.
Visitez https://ubuntu.com/esm ou executez : sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Mon Apr 14 15:21:31 2025 from 192.168.56.101
```

## 4. Environnement et outils

- **OS** : Linux Ubuntu pour les deux VMs
- **Client SSH** : ligne de commande Linux
- **Serveur SSH** : OpenSSH
- **Machines** :
  - Serveur : Serveur Ubuntu, IP : 192.168.56.102

- Client : Client Ubuntu, IP : 192.168.56.101

## 5. Analyse et conclusion

- L'utilisation du mot de passe permet une connexion simple mais peu sécurisée.
- Désactiver l’authentification par mot de passe empêche les connexions non autorisées.
- L’authentification par clés est une méthode plus sûre et recommandée en environnement professionnel.

Ce TP nous a permis de mieux comprendre la configuration de base d'un accès SSH sécurisé et les bonnes pratiques à adopter dans un contexte réel d'administration réseau.

# Quatrième TP : Administration SSH (étape 3 et 4)

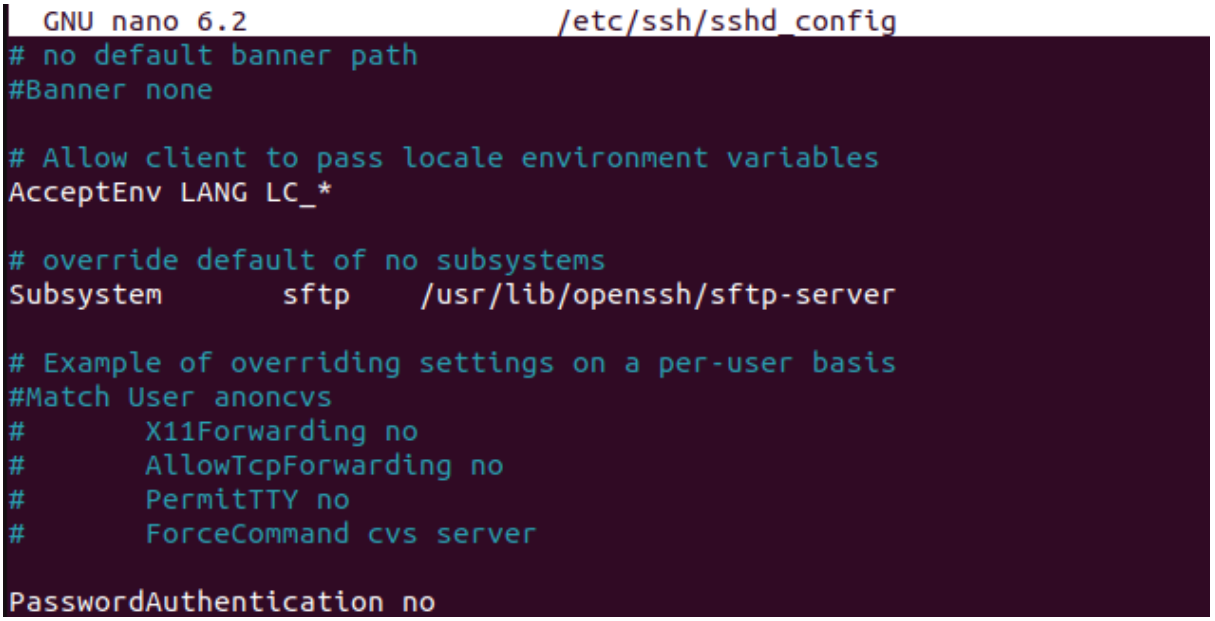
## Étape 3 – Optimisation de la configuration du serveur SSH

### Objectif

Renforcer la sécurité de l’accès SSH en modifiant le fichier de configuration du serveur.

### Optimisations mises en place

### Désactivation de l’authentification par mot de passe




Cela permet d’éviter toute tentative de connexion par bruteforce via mot de passe.

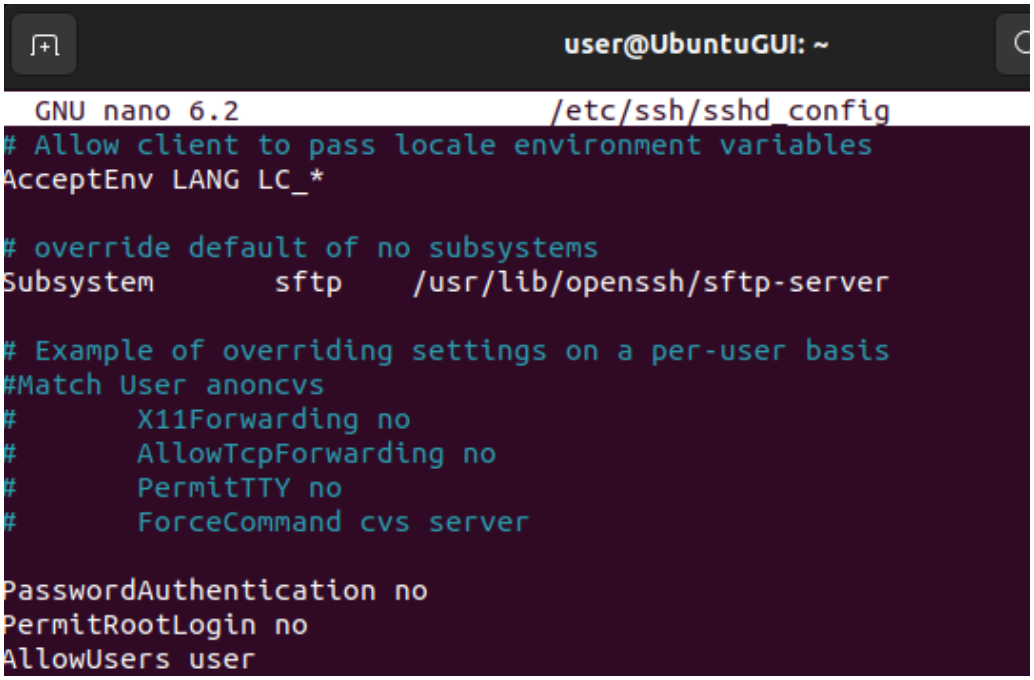
Voici le test après modification du port :

## Blocage de l'accès SSH au compte root

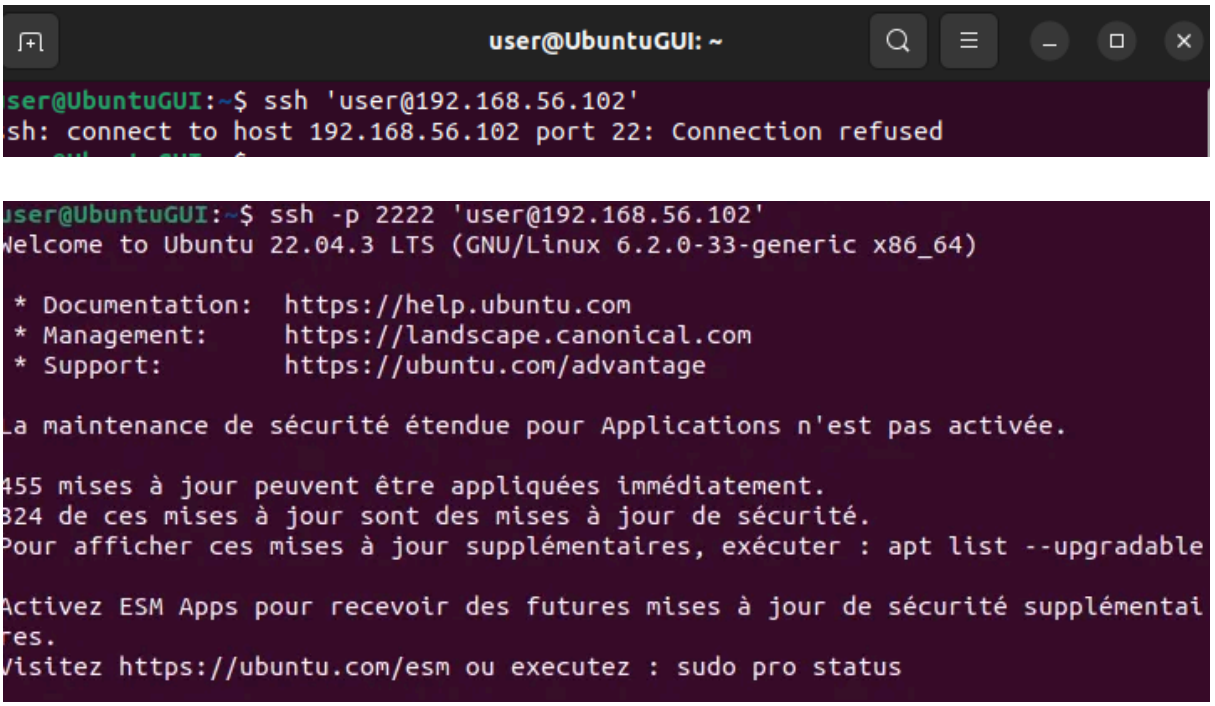
Permet d’éviter qu’un attaquant cible directement le superutilisateur.

## Restriction à des utilisateurs spécifiques

Pour limiter les utilisateurs on fait en sorte de mettre la commande AllowUsers (notre/nos utilisateur(s)) et ainsi refuser toutes les entrées externes.




Voici un test avant et après avoir autorisé notre “user” :




## Tests effectués

- Connexion en root refusée
- Connexion par mot de passe refusée
- Connexion avec clé sur le port personnalisé

- Tentative de connexion par utilisateur non listé dans `AllowUsers` refusée

# Cinquième TP : Syslog

Serveur :

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

```
$template RemoteLogs,"/var/log/remote/%HOSTNAME%.log"
*.* ?RemoteLogs
```

Client :

```
#input(type="imudp" port="514")
*.* @192.168.56.101
# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")
*.* @@192.168.56.101
```

Envoie d'un test sur le client :

```
user@UbuntuGUI:~$ logger "Message test"
```

Réception sur le serveur :

```
user@UbuntuGUI:~$ sudo tail -f /var/log/remote/*.log
May 26 17:23:55 UbuntuGUI systemd[1262]: app-gnome-gnome\x2dnetwork\x2dpanel-29
6.scope: Consumed 4.016s CPU time.
May 26 17:23:56 UbuntuGUI avahi-daemon[560]: Server startup complete. Host name
is UbuntuGUI-4.local. Local service cookie is 3912303592.
May 26 17:24:04 UbuntuGUI systemd[1]: NetworkManager-dispatcher.service: Deacti
ated successfully.
May 26 17:24:04 UbuntuGUI user: Message test
May 26 17:24:04 UbuntuGUI rsyslogd: action 'action-1-builtin:omfwd' resumed (mo
ule 'builtin:omfwd') [v8.2112.0 try https://www.rsyslog.com/e/2359 ]
May 26 17:24:04 UbuntuGUI user: Message test
May 26 17:24:04 UbuntuGUI rsyslogd: action 'action-1-builtin:omfwd' resumed (mo
ule 'builtin:omfwd') [v8.2112.0 try https://www.rsyslog.com/e/2359 ]
May 26 17:24:05 UbuntuGUI systemd[1]: systemd-hostnamed.service: Deactivated su
cessfully.
```

Test d’erreur :

```
user@UbuntuGUI:~$ sudo cat /var/log/auth-errors.log
May 26 17:35:57 UbuntuGUI user: Test erreur
May 26 17:35:57 UbuntuGUI user: Test erreur
```

# Sixième TP : Les scanners

## Objectif

Découvrir un outil simple et rapide pour scanner les hôtes d’un réseau local et détecter les adresses IP actives, ainsi que les ports ouverts.

---

## Installation sur Ubuntu :

Pour installer **Angry IP Scanner** depuis GitHub :

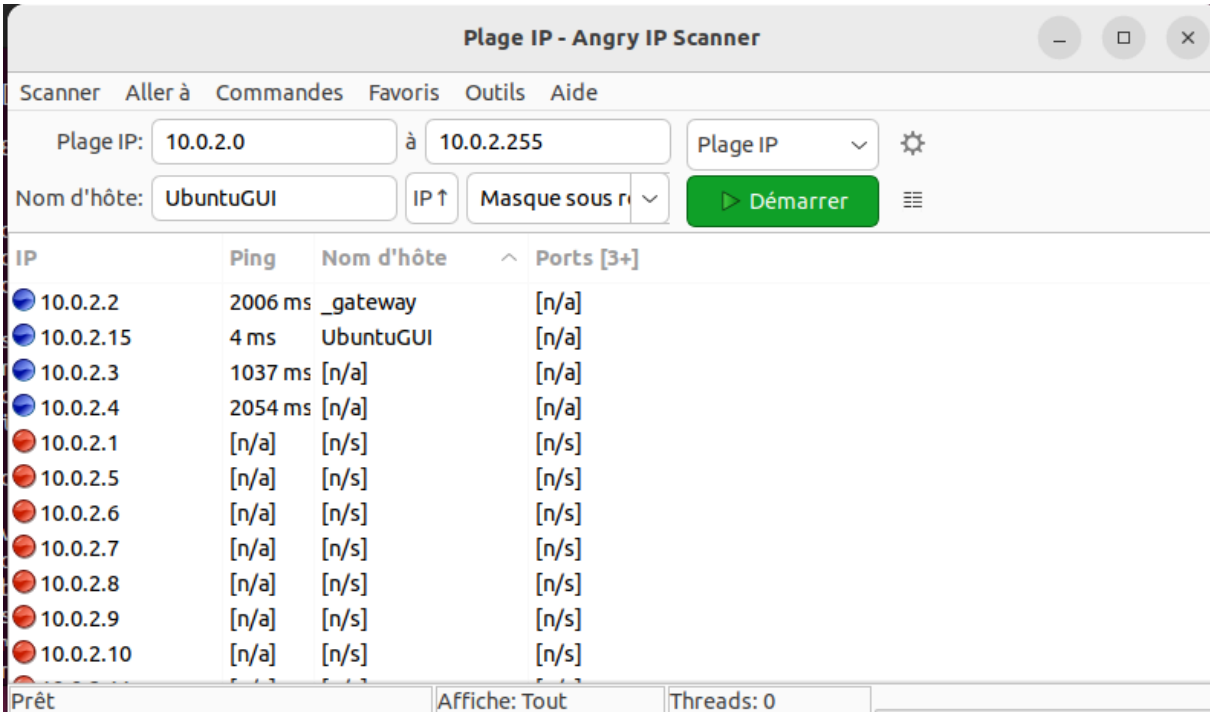
cd /tmp
wget
https://github.com/angryip/ipscan/releases/download/3.9.1/ipscan_3.9.1_amd64.deb
sudo apt install ./ipscan_3.9.1_amd64.deb

```
user@UbuntuGUI:~$ cd /tmp
user@UbuntuGUI:/tmp$ wget https://github.com/angryip/ipscan/releases/download/3.
9.1/ipscan_3.9.1_amd64.deb
--2025-06-02 14:20:38--  https://github.com/angryip/ipscan/releases/download/3.9
.1/ipscan_3.9.1_amd64.deb
```

```
user@UbuntuGUI:/tmp$ sudo apt install ./ipscan_3.9.1_amd64.deb
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Rhythmbox ction de « ipscan » au lieu de « ./ipscan_3.9.1_amd64.deb »
Les paquets supplémentaires suivants seront installés :
```

## Lancement

Après installation, le logiciel peut être lancé graphiquement, ou bien par commande dans le terminal avec : ipscan




## Test réalisé

- **Plage scannée** : `10.0.2.0` → `10.0.2.255`
- **Informations détectées** :
  - IP actives
  - Nom d'hôte
  - Réponse au ping
  - Ports TCP ouverts

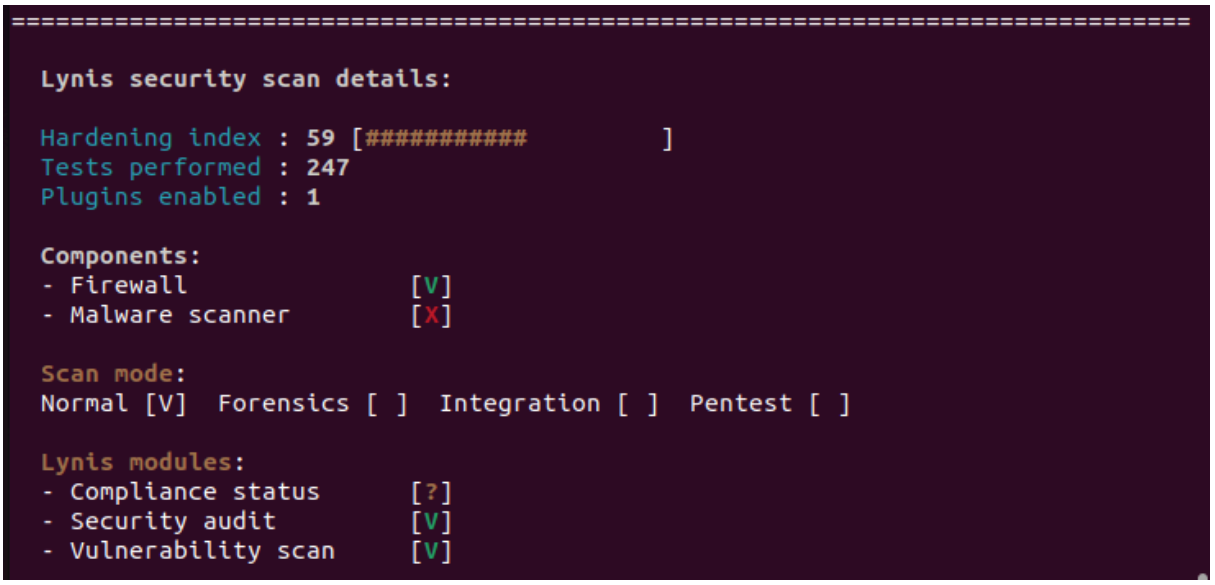
## Résultats

Angry IP Scanner a détecté plusieurs machines en ligne sur le réseau. Il a permis d'avoir une **vue rapide de la topologie active**, utile pour un premier audit ou une détection de machines non autorisées.

## Installation sur Ubuntu :

sudo apt install lynis

sudo lynis audit system




## Test réalisé

- Lynis a analysé plusieurs éléments de sécurité :
  - Permissions de fichiers système critiques
  - Services actifs
  - Configuration réseau
  - Utilisation du pare-feu
  - Authentification, mots de passe, sudo
  - Mises à jour de sécurité
  - Audit de logs et journaux

### Conclusion

Lynis est un outil incontournable pour auditer rapidement un système Linux et **repérer les points faibles de configuration**. C'est un **excellent point de départ**

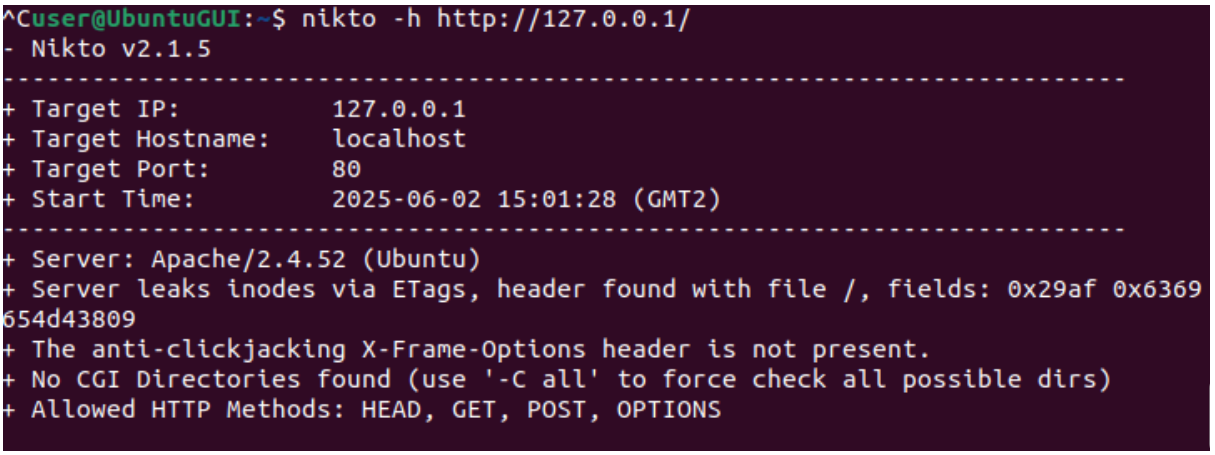
pour renforcer la sécurité avant d'installer d'autres services sensibles ou exposés.

## Installation sur Ubuntu :

sudo apt install nikto

sudo apt install apache2

nikto -h http://127.0.0.1/




## Test réalisé

- **Cible scannée** : `http://127.0.0.1` (le serveur Apache local tournant sur la machine)
- **Type de scan** : Analyse standard avec détection de failles connues et de fichiers critiques

## Résultats observés

Nikto a effectué plusieurs vérifications sur la cible :

- Présence de fichiers sensibles (`robots.txt`, `phpinfo()`, `server-status`, etc.)
- Vérification de la version du serveur web (Apache, nginx, etc.)
- Recherches de failles connues (ex : CVE)
- Analyse des en-têtes HTTP

Le scan a permis de détecter :

- D'éventuels points faibles ou fichiers exposés

- Les méthodes HTTP activées
- L’absence ou la présence d’informations serveur divulguées dans les en-têtes

### Conclusion

Nikto est un outil très utile pour **identifier rapidement les erreurs courantes** de configuration et les fichiers exposés sur un serveur web. C’est une étape indispensable dans une **analyse de surface d’un service HTTP**.

## Conclusion Générale du TP – Scanners de Sécurité

Ce travail pratique a permis de découvrir et d’expérimenter différents outils de **scan de sécurité** couvrant les domaines du **réseau**, du **système** et des **applications**.

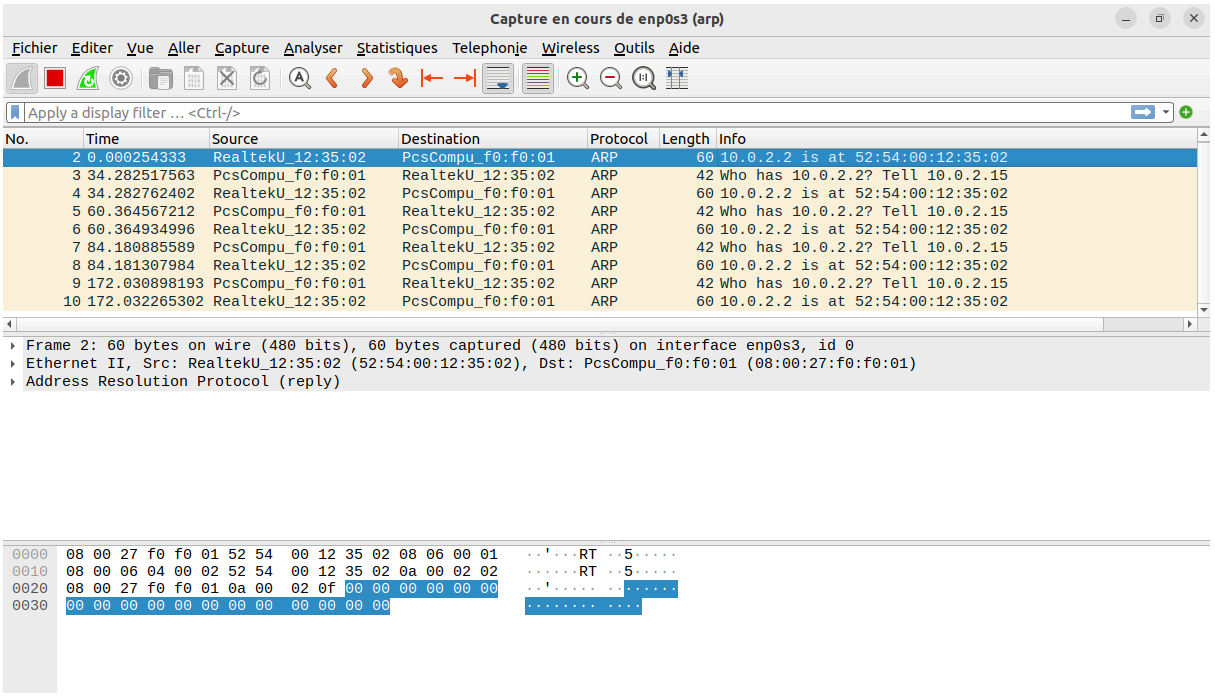
- **Lynis** a réalisé un **audit approfondi du système Linux**, mettant en évidence les configurations sensibles, les services actifs ainsi que des recommandations pour renforcer la sécurité.
- **Nikto** a été utilisé pour **scanner un serveur web local**, en identifiant des vulnérabilités connues, des fichiers à risque et des défauts de configuration HTTP.
- **Angry IP Scanner** a permis de **scanner rapidement un réseau local**, en détectant les adresses IP actives et les ports ouverts.

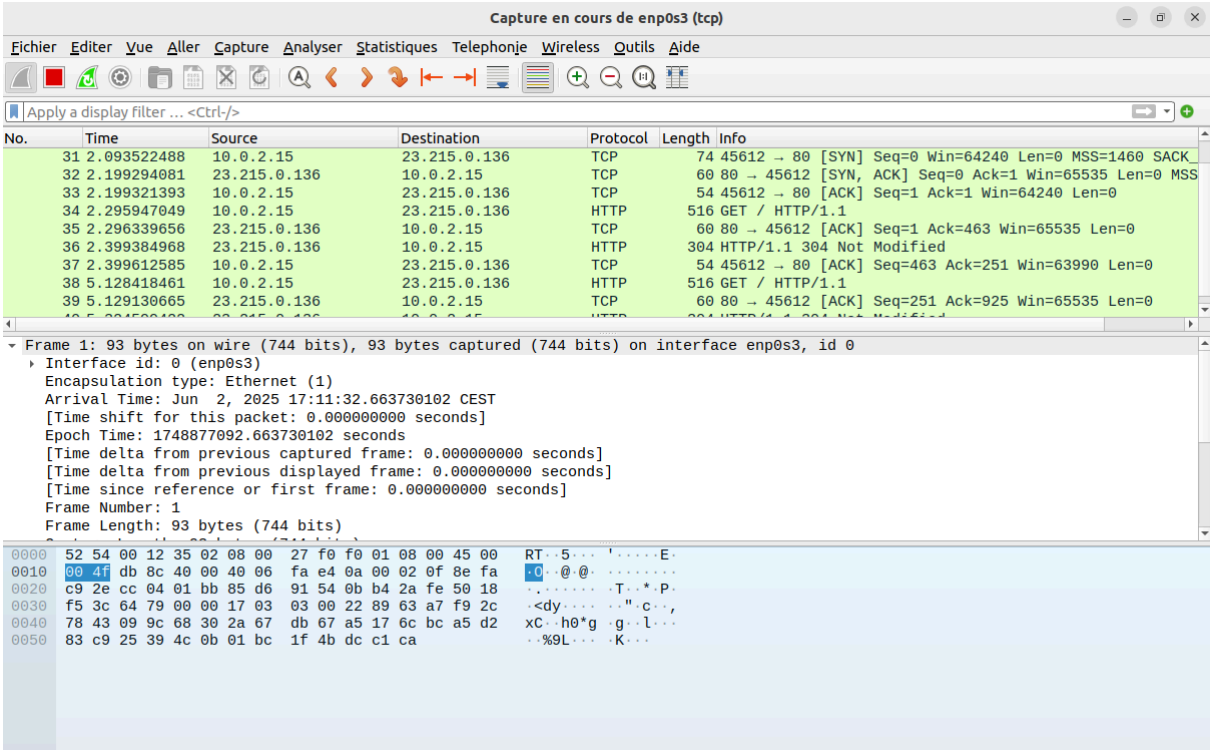
Ces outils sont **complémentaires** et montrent bien l’intérêt de la supervision à plusieurs niveaux :

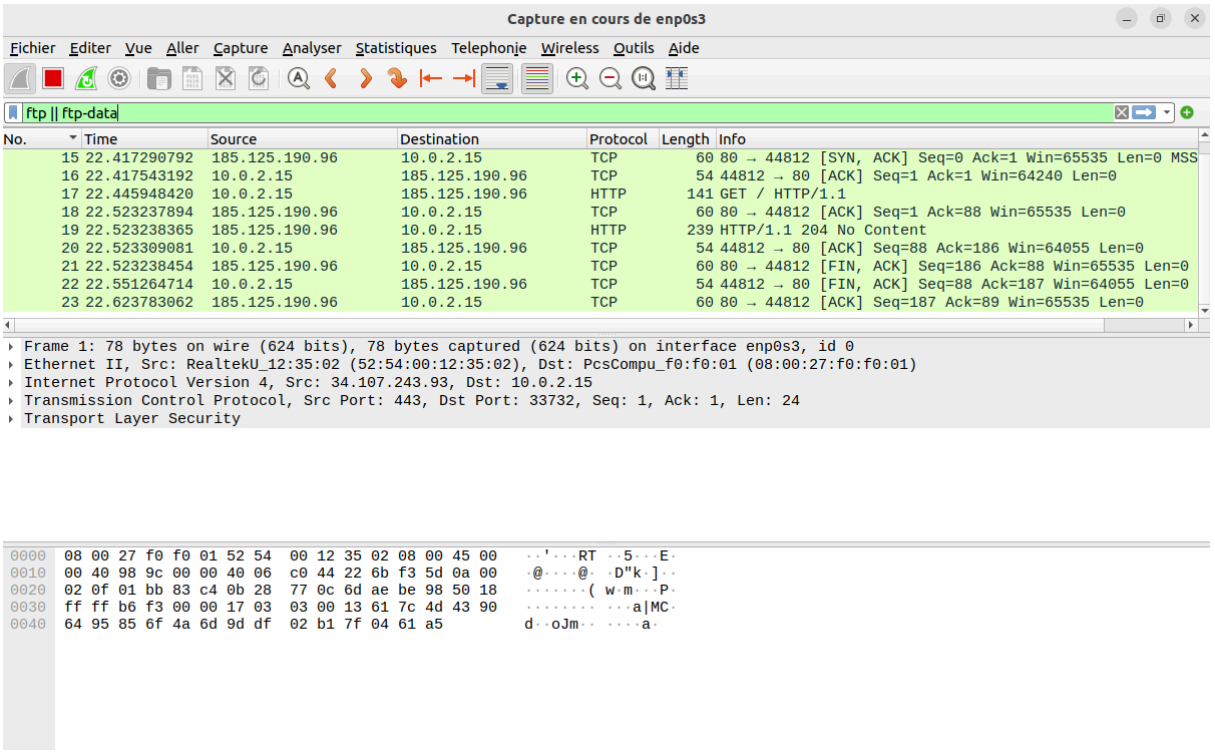
- Le **réseau**, pour surveiller les points d’entrée accessibles.
- Les **applications**, pour détecter des failles exploitables.
- Le **système**, pour s’assurer d’un bon niveau de configuration et de sécurité.

Ce TP met en évidence l’importance des scanners comme outils de **détection préventive**, permettant d’identifier rapidement des faiblesses dans une infrastructure avant qu’elles ne soient exploitées.

# Septième TP : Analyse réseau


Capture en cours de enp0s3 (arp)
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
2 0.000254333 RealtekU_12:35:02 PcsCompu_f0:f0:01 ARP 60 10.0.2.2 is at 52:54:00:12:35:02
3 34.282517563 PcsCompu_f0:f0:01 RealtekU_12:35:02 ARP 42 Who has 10.0.2.2? Tell 10.0.2.15
4 34.282762402 RealtekU_12:35:02 PcsCompu_f0:f0:01 ARP 60 10.0.2.2 is at 52:54:00:12:35:02
5 60.364567212 PcsCompu_f0:f0:01 RealtekU_12:35:02 ARP 42 Who has 10.0.2.2? Tell 10.0.2.15
6 60.364934996 RealtekU_12:35:02 PcsCompu_f0:f0:01 ARP 60 10.0.2.2 is at 52:54:00:12:35:02
7 84.180885589 PcsCompu_f0:f0:01 RealtekU_12:35:02 ARP 42 Who has 10.0.2.2? Tell 10.0.2.15
8 84.181307984 RealtekU_12:35:02 PcsCompu_f0:f0:01 ARP 60 10.0.2.2 is at 52:54:00:12:35:02
9 172.030898193 PcsCompu_f0:f0:01 RealtekU_12:35:02 ARP 42 Who has 10.0.2.2? Tell 10.0.2.15
10 172.032265302 RealtekU_12:35:02 PcsCompu_f0:f0:01 ARP 60 10.0.2.2 is at 52:54:00:12:35:02
Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_f0:f0:01 (08:00:27:f0:f0:01)
Address Resolution Protocol (reply)
0000 08 00 27 f0 f0 01 52 54 00 12 35 02 08 06 00 01
0010 08 00 06 04 00 02 52 54 00 12 35 02 0a 00 02 02
0020 08 00 27 f0 f0 01 0a 00 02 0f 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00


Capture en cours de enp0s3 (tcp)
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
31 2.093522488 10.0.2.15 23.215.0.136 TCP 74 45612 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_
32 2.199294081 23.215.0.136 10.0.2.15 TCP 60 80 → 45612 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS
33 2.199321393 10.0.2.15 23.215.0.136 TCP 54 45612 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
34 2.295947049 10.0.2.15 23.215.0.136 HTTP 516 GET / HTTP/1.1
35 2.296339656 23.215.0.136 10.0.2.15 TCP 60 80 → 45612 [ACK] Seq=1 Ack=463 Win=65535 Len=0
36 2.399384968 23.215.0.136 10.0.2.15 HTTP 304 HTTP/1.1 304 Not Modified
37 2.399612585 10.0.2.15 23.215.0.136 TCP 54 45612 → 80 [ACK] Seq=463 Ack=251 Win=63990 Len=0
38 5.128418461 10.0.2.15 23.215.0.136 HTTP 516 GET / HTTP/1.1
39 5.129130665 23.215.0.136 10.0.2.15 TCP 60 80 → 45612 [ACK] Seq=251 Ack=925 Win=65535 Len=0
Frame 1: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface enp0s3, id 0
Interface id: 0 (enp0s3)
Encapsulation type: Ethernet (1)
Arrival Time: Jun 2, 2025 17:11:32.663730102 CEST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1748877092.663730102 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 93 bytes (744 bits)
0000 52 54 00 12 35 02 08 00 27 f0 f0 01 08 00 45 00
0010 00 4f db 8c 40 00 40 06 fa e4 0a 00 02 0f 8e fa
0020 c9 2e cc 04 01 bb 85 d6 91 54 0b b4 2a fe 50 18
0030 f5 3c 64 79 00 00 17 03 03 00 22 89 63 a7 f9 2c
0040 78 43 09 9c 68 30 2a 67 db 67 a5 17 6c bc a5 d2
0050 83 c9 25 39 4c 0b 01 bc 1f 4b dc c1 ca


Capture en cours de enp0s3
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
ftp || ftp-data
No. Time Source Destination Protocol Length Info
15 22.417290792 185.125.190.96 10.0.2.15 TCP 60 80 → 44812 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS
16 22.417543192 10.0.2.15 185.125.190.96 TCP 54 44812 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
17 22.445948420 10.0.2.15 185.125.190.96 HTTP 141 GET / HTTP/1.1
18 22.523237894 185.125.190.96 10.0.2.15 TCP 60 80 → 44812 [ACK] Seq=1 Ack=88 Win=65535 Len=0
19 22.523238365 185.125.190.96 10.0.2.15 HTTP 239 HTTP/1.1 204 No Content
20 22.523309081 10.0.2.15 185.125.190.96 TCP 54 44812 → 80 [ACK] Seq=88 Ack=186 Win=64055 Len=0
21 22.523238454 185.125.190.96 10.0.2.15 TCP 60 80 → 44812 [FIN, ACK] Seq=186 Ack=88 Win=65535 Len=0
22 22.551264714 10.0.2.15 185.125.190.96 TCP 54 44812 → 80 [FIN, ACK] Seq=88 Ack=187 Win=64055 Len=0
23 22.623783062 185.125.190.96 10.0.2.15 TCP 60 80 → 44812 [ACK] Seq=187 Ack=89 Win=65535 Len=0
Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface enp0s3, id 0
Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_f0:f0:01 (08:00:27:f0:f0:01)
Internet Protocol Version 4, Src: 34.107.243.93, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 33732, Seq: 1, Ack: 1, Len: 24
Transport Layer Security
0000 08 00 27 f0 f0 01 52 54 00 12 35 02 08 00 45 00
0010 00 40 98 9c 00 00 40 06 c0 44 22 6b f3 5d 0a 00
0020 02 0f 01 bb 83 c4 0b 28 77 0c 6d ae be 98 50 18
0030 ff ff b6 f3 00 00 17 03 03 00 13 61 7c 4d 43 90
0040 64 95 85 6f 4a 6d 9d df 02 b1 7f 04 61 a5