Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

COMPTE RENDU

Cyber Sécurité - TP1 - Propriétés de Sécurité - Triade CIA
3e année Cybersécurité - École Supérieure d'Informatique et du Numérique (ESIN)
Collège d'Ingénierie & d'Architecture (CIA)

Étudiants : HATHOUTI Mohammed Taha
IRAQUI HOUSSAINI Ghali

Filière : Cybersecurité

Année : 2025/2026

Enseignant : M.SEBBAR & Mme.GADI

Date : 26 janvier 2026

Table des matières

Objectifs

L'objectif de ce TP est de mettre en place un réseau entre différentes machines et d'analyser le trafic réseau et les connexions client-serveur.

Dans ce TP, nous nous concentrerons sur :

— La capture et l'analyse du trafic du protocole FTP ;

— La capture et l'analyse du trafic du protocole Telnet ;

— La capture et l'analyse du trafic du protocole SSH ;

Instructions

— Le rapport de TP doit être soumis une semaine après la séance de TP sur la plateforme Moodle, en respectant la date limite mentionnée ;

— Le TP doit être réalisé individuellement en classe, mais le rapport doit être soumis en groupes de 2 étudiants maximum ;

— Les groupes de TP doivent rester les mêmes pour tous les rapports tout au long du semestre ;

Propriétés de Sécurité (Rappel)

Identifiez quelle propriété de sécurité (Confidentialité, Intégrité, Disponibilité) est violée dans chacun des cas suivants :

1. Un pirate attaque une usine pharmaceutique et modifie la formule chimique des médicaments produits : **Intégrité**
2. Un pirate vole la formule chimique d'un médicament particulier : **Confidentialité**
3. Un pirate effectue une attaque DDoS sur un serveur de messagerie : **Disponibilité**
4. Un pirate exécute une violation de données sur un serveur de données : **Confidentialité**
5. Une entreprise médiatique subit une attaque par ransomware : **Confidentialité & Disponibilité**

Configuration du Réseau

Configuration de la Machine Windows

Pour ce TP, nous avons connecté deux machines via un câble Ethernet :

— Machine Windows 11 (Client) : IP = **192.168.1.10/24**

— Machine Ubuntu (Serveur) : IP = **192.168.1.20/24**

Configuration IP Windows

Sur la machine Windows, nous avons configuré l'adresse IP statique via la commande :

```
netsh interface ip set address name="Ethernet" source=static addr
   =192.168.1.10 mask=255.255.255.0
```

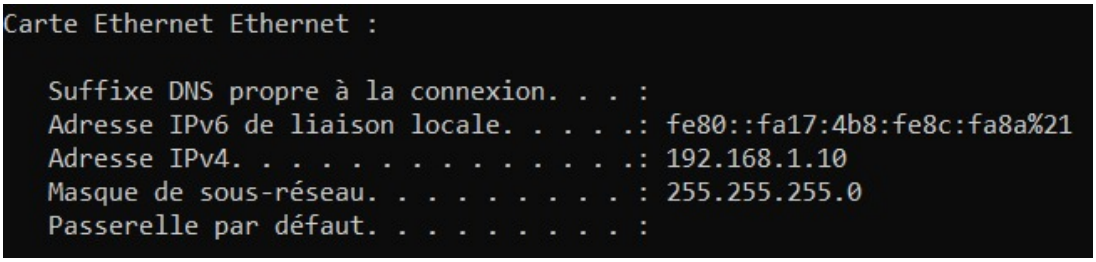


FIGURE 1 – Configuration de l'adresse IP Windows via netsh

Configuration de la Machine Ubuntu

Sur la machine Ubuntu, nous avons configuré l'interface Ethernet avec l'adresse IP 192.168.1.20 :

```
sudo ifconfig enx0c3796401539 192.168.1.20 netmask 255.255.255.0
   up
```

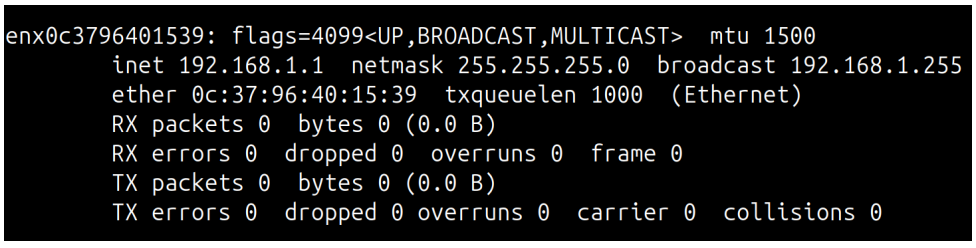


FIGURE 2 – Configuration de l'interface réseau Ubuntu avec ifconfig

Test de Connectivité

Nous avons vérifié la connectivité entre les deux machines avec la commande ping :

```
hathouti@Taha-inspiron-16:~$ ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=128 time=2.74 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=128 time=1.69 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=128 time=1.75 ms
64 bytes from 192.168.1.10: icmp_seq=4 ttl=128 time=2.51 ms
 64 bytes from 192.168.1.10: icmp_seq=5 ttl=128 time=1.79 ms
^C
--- 192.168.1.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 1.687/2.095/2.742/0.440 ms
```

FIGURE 3 – Test de connectivité avec ping depuis Ubuntu vers Windows (192.168.1.10)

La connectivité est bien établie avec un taux de perte de 0%.

1 Capture et Analyse du Trafic du Protocole FTP

1.1 Processus de Capture

1.1.1 Installation et Démarrage du Service FTP

Sur la machine Ubuntu, nous avons installé et démarré le serveur FTP vsftpd :

```
sudo apt update
sudo apt install vsftpd -y
sudo service vsftpd start
```

```
hathouti@Taha-inspiron-16:~$ service vsftpd status
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-01-26 15:00:13 +01; 17min ago
    Process: 76017 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
   Main PID: 76019 (vsftpd)
      Tasks: 1 (limit: 37952)
     Memory: 752.0K (peak: 1.5M)
        CPU: 13ms
     CGroup: /system.slice/vsftpd.service
             └─76019 /usr/sbin/vsftpd /etc/vsftpd.conf

janv. 26 15:00:13 Taha-inspiron-16 systemd[1]: Starting vsftpd.service - vsftpd FTP server...
janv. 26 15:00:13 Taha-inspiron-16 systemd[1]: Started vsftpd.service - vsftpd FTP server.
```

FIGURE 4 – Statut du service vsftpd - Service actif et en cours d'exécution

1.1.2 À quelle couche du modèle OSI appartient le protocole FTP ? Quel est le numéro de port utilisé par FTP ?

- **Couche OSI :** Le protocole FTP appartient à la **couche Application (couche 7)** du modèle OSI.
- **Ports utilisés :**
  - Port **21** : Port de contrôle (commandes FTP)
  - Port **20** : Port de données (transfert de fichiers)

1.1.3 Connexion FTP depuis Windows

Depuis la machine Windows, nous avons établi une connexion FTP vers le serveur Ubuntu :

```
ftp 192.168.1.20
```

Identifiants utilisés :

— Utilisateur : **hathouti**

— Mot de passe : **test-2026**

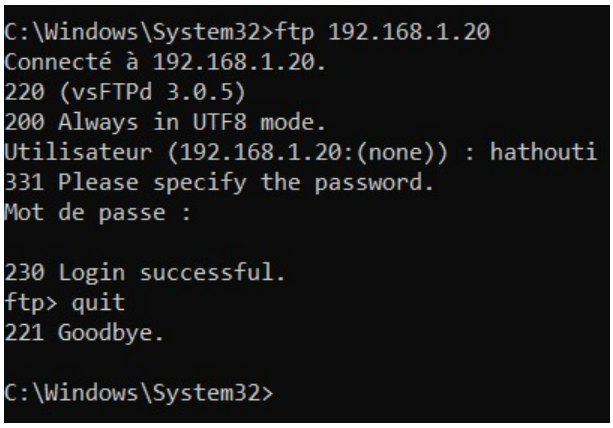


FIGURE 5 – Connexion FTP réussie depuis Windows vers Ubuntu

La connexion FTP a été établie avec succès. Le message "230 Login successful" confirme l'authentification réussie.

1.2 Analyse du Trafic FTP

1.2.1 Quel protocole de la couche transport FTP utilise-t-il ?

FTP utilise le protocole **TCP (Transmission Control Protocol)** comme protocole de transport. TCP garantit une connexion fiable et ordonnée, ce qui est essentiel pour le transfert de fichiers.

No.	Time	Source	Destination	Protocol	Length	Info
19	19.462129	192.168.1.20	192.168.1.10	TCP	60	21 → 50831 [FIN, ACK] Seq=118 Ack=52 Win=64256 Len=0
20	19.462262	192.168.1.10	192.168.1.20	TCP	54	50831 → 21 [ACK] Seq=52 Ack=119 Win=8075 Len=0
21	19.469821	192.168.1.10	192.168.1.20	TCP	54	50831 → 21 [FIN, ACK] Seq=52 Ack=119 Win=8075 Len=0
22	19.472109	192.168.1.20	192.168.1.10	TCP	60	21 → 50831 [ACK] Seq=119 Ack=53 Win=64256 Len=0
23	53.503863	192.168.1.10	192.168.1.20	TCP	66	56326 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
24	53.505576	192.168.1.20	192.168.1.10	TCP	66	21 → 56326 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
25	53.505686	192.168.1.10	192.168.1.20	TCP	54	56326 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0
26	53.511079	192.168.1.20	192.168.1.10	FTP	74	Response: 220 (vsFTPd 3.0.5)
27	53.525309	192.168.1.10	192.168.1.20	FTP	68	Request: OPTS UTF8 ON
28	53.526971	192.168.1.20	192.168.1.10	TCP	60	21 → 56326 [ACK] Seq=21 Ack=15 Win=64256 Len=0
29	53.526971	192.168.1.20	192.168.1.10	FTP	80	Response: 200 Always in UTF8 mode.
30	53.581464	192.168.1.10	192.168.1.20	TCP	54	56326 → 21 [ACK] Seq=15 Ack=47 Win=8146 Len=0
31	58.400270	192.168.1.10	192.168.1.20	FTP	69	Request: USER hathouti
32	58.401469	192.168.1.20	192.168.1.10	FTP	88	Response: 331 Please specify the password.
33	58.453351	192.168.1.10	192.168.1.20	TCP	54	56326 → 21 [ACK] Seq=30 Ack=81 Win=8112 Len=0
34	62.974136	192.168.1.10	192.168.1.20	FTP	70	Request: PASS test-2026
35	63.015285	192.168.1.20	192.168.1.10	TCP	60	21 → 56326 [ACK] Seq=81 Ack=46 Win=64256 Len=0
36	63.058551	192.168.1.20	192.168.1.10	FTP	77	Response: 230 Login successful.
37	63.102736	192.168.1.10	192.168.1.20	TCP	54	56326 → 21 [ACK] Seq=46 Ack=104 Win=8089 Len=0
38	71.470321	192.168.1.20	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
39	72.471488	192.168.1.20	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
40	73.471922	192.168.1.20	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
41	74.472864	192.168.1.20	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
42	82.250645	192.168.1.10	192.168.1.20	FTP	60	Request: QUIT
43	82.252889	192.168.1.20	192.168.1.10	TCP	60	21 → 56326 [ACK] Seq=104 Ack=52 Win=64256 Len=0
44	82.252889	192.168.1.20	192.168.1.10	FTP	68	Response: 221 Goodbye.

FIGURE 6 – Capture Wireshark montrant les paquets FTP utilisant TCP

1.2.2 Identifiez les messages échangés entre le client FTP et le serveur. Quels messages sont utilisés pour établir la connexion TCP initiale ?

L'établissement de la connexion TCP utilise le **three-way handshake** :

1. **SYN** : Le client envoie un paquet SYN au serveur pour initier la connexion ;
2. **SYN-ACK** : Le serveur répond avec un paquet SYN-ACK pour accepter la connexion ;
3. **ACK** : Le client envoie un ACK final pour confirmer l'établissement de la connexion ;

Dans la capture Wireshark, on observe :

— Paquet #23 : [SYN] du client (192.168.1.10) vers le serveur (192.168.1.20)

— Paquet #24 : [SYN, ACK] du serveur vers le client

— Paquet #25 : [ACK] du client pour confirmer la connexion

Ensuite, les commandes FTP sont échangées :

— Response : 220 (Code de bienvenue vsftpd)

— Request : USER hathouti

— Response : 331 Please specify the password

— Request : PASS (mot de passe)

— Response : 230 Login successful

1.2.3 Quelles sont les valeurs des ports source et destination ? Quel est leur objectif ?

Dans la capture, on observe :

— **Port destination (serveur)** : **21** - Port FTP standard pour les commandes (fixe) ;

— **Port source (client)** : **57818** - Port éphémère choisi dynamiquement (variable) ;

```
hathouti@Taha-inspiron-16:~$ netstat -an | grep :21
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 192.168.1.20:21         192.168.1.10:57818      ESTABLISHED
```

FIGURE 7 – Observation avec netstat - Une seule connexion FTP

Objectif des ports :

— Le **port 21** est un port bien connu réservé pour le service FTP. Il permet aux clients de savoir où se connecter ;

— Le **port source** (éphémère) est choisi aléatoirement par le système d'exploitation client pour identifier la connexion de manière unique ;

1.2.4 Si vous ouvrez une nouvelle connexion FTP, les numéros de port changent-ils ?

Oui, le port source côté client change à chaque nouvelle connexion.

Observation avec netstat :

```
hathouti@Taha-inspiron-16:~$ netstat -an | grep :21
tcp6       0      0 ::21                    :::*                    LISTEN
tcp6       0      0 192.168.1.20:21         192.168.1.10:57818      ESTABLISHED
hathouti@Taha-inspiron-16:~$ netstat -an | grep :21
tcp6       0      0 ::21                    :::*                    LISTEN
tcp6       0      0 192.168.1.20:21         192.168.1.10:50831      TIME_WAIT
hathouti@Taha-inspiron-16:~$ netstat -an | grep :20
hathouti@Taha-inspiron-16:~$ netstat -an | grep :21
tcp6       0      0 ::21                    :::*                    LISTEN
tcp6       0      0 192.168.1.20:21         192.168.1.10:56326      ESTABLISHED
tcp6       0      0 192.168.1.20:21         192.168.1.10:50831      TIME_WAIT
hathouti@Taha-inspiron-16:~$ netstat -an | grep :21
tcp6       0      0 ::21                    :::*                    LISTEN
tcp6       0      0 192.168.1.20:21         192.168.1.10:56326      TIME_WAIT
```

FIGURE 8 – Observation avec netstat - Plusieurs connexions FTP avec des ports différents

On observe plusieurs connexions FTP :

— Connexion 1 : 192.168.1.20 :21 ↔ 192.168.1.10 :**57818** (ESTABLISHED)

— Connexion 2 : 192.168.1.20 :21 ↔ 192.168.1.10 :**50831** (TIME_WAIT)

— Connexion 3 : 192.168.1.20 :21 ↔ 192.168.1.10 :**56326** (ESTABLISHED)

Explication :

— Le **port serveur (21)** reste constant

— Le **port client** change pour chaque nouvelle connexion (57818, 50831, 56326...)

— Cela permet au système d'exploitation de gérer plusieurs connexions simultanées

1.2.5 Analysez les autres messages échangés, en particulier les messages TCP

Dans la capture Wireshark, on observe plusieurs types de messages TCP :

— **[FIN, ACK]** : Indique la fermeture propre de la (Paquet #21) ;

— **[ACK]** : Accusés de réception pour chaque paquet de données (Paquet #22) ;

— **Messages FTP** :

  — Request : QUIT - Demande de déconnexion (Paquet #42) ;

  — Response : 221 Goodbye - Confirmation de déconnexion (Paquet #44) ;

Constat de sécurité :

— Les identifiants (login/mot de passe) sont visibles en clair dans Wireshark ;

— Toutes les commandes exécutées sont interceptables ;

2 Capture et Analyse du Trafic du Protocole Telnet

2.1 Processus de Capture

2.1.1 Installation et Démarrage du Service Telnet

Sur la machine Ubuntu, nous avons installé et configuré le service Telnet avec xinetd :

```
sudo apt install xinetd telnetd
sudo nano /etc/xinetd.d/telnet
```

Configuration dans `/etc/xinetd.d/telnet` :

```
service telnet
{
    disable = no
    flags = REUSE
    socket_type = stream
    wait = no
    user = root
    server = /usr/sbin/telnetd
    log_on_failure += USERID
}
```

```
sudo service xinetd restart
```

```
hathouti@Taha-inspiron-16:~$ sudo netstat -tulpn | grep :23
tcp6       0      0 :::23                   :::*                    LISTEN      82740/xinetd
```

FIGURE 9 – Vérification du port 23 (Telnet) avec netstat - Service en écoute

2.1.2 À quelle couche du modèle OSI appartient le protocole Telnet ? Quel est son numéro de port ?

— **Couche OSI :** Telnet appartient à la **couche Application (couche 7)** du modèle OSI.

— **Port utilisé :** Port **23** (port standard pour Telnet)

2.1.3 Connexion Telnet depuis Windows

Depuis la machine Windows, nous avons établi une connexion Telnet vers le serveur Ubuntu :

```
telnet 192.168.1.20
```

Identifiants utilisés :

— Utilisateur : **hathouti**

— Mot de passe : **test-2026**

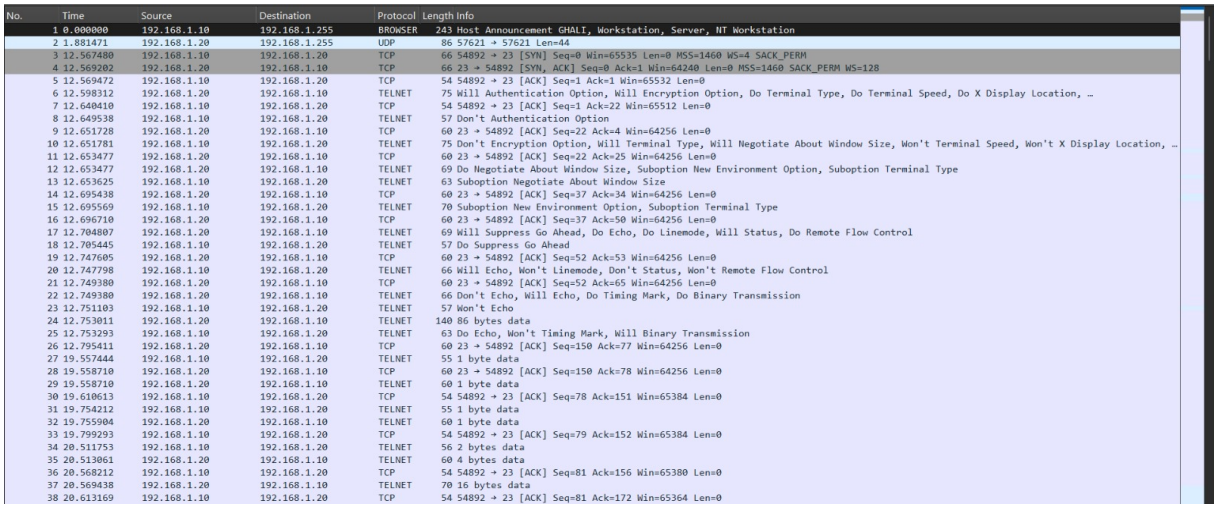
FIGURE 10 – Connexion Telnet réussie et exécution de commandes (ls)

La connexion Telnet a été établie avec succès. Nous pouvons voir l'écran de bienvenue Ubuntu et l'exécution de la commande `ls`.

2.2 Analyse du Trafic Telnet

2.2.1 Quel protocole de la couche transport Telnet utilise-t-il ?

Telnet utilise également le protocole **TCP (Transmission Control Protocol)** pour assurer une communication fiable et ordonnée.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.10	192.168.1.255	BROWSER	243	Host Announcement GHALI, Workstation, Server, NT Workstation
2	1.881471	192.168.1.20	192.168.1.255	UDP	86	57621 → 57621 Len=44
3	12.567480	192.168.1.10	192.168.1.20	TCP	66	54892 → 23 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=4 SACK_PERM
4	12.569202	192.168.1.20	192.168.1.10	TCP	66	23 → 54892 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	12.569472	192.168.1.10	192.168.1.20	TCP	54	54892 → 23 [ACK] Seq=1 Ack=1 Win=65532 Len=0
6	12.598312	192.168.1.20	192.168.1.10	TELNET	75	Will Authentication Option, Will Encryption Option, Do Terminal Type, Do Terminal Speed, Do X Display Location, …
7	12.640410	192.168.1.10	192.168.1.20	TCP	54	54892 → 23 [ACK] Seq=1 Ack=22 Win=65512 Len=0
8	12.649538	192.168.1.10	192.168.1.20	TELNET	57	Don't Authentication Option
9	12.651728	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=22 Ack=4 Win=64256 Len=0
10	12.651781	192.168.1.10	192.168.1.20	TELNET	75	Don't Encryption Option, Will Terminal Type, Will Negotiate About Window Size, Won't Terminal Speed, Won't X Display Location, …
11	12.653477	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=22 Ack=25 Win=64256 Len=0
12	12.653477	192.168.1.20	192.168.1.10	TELNET	69	Do Negotiate About Window Size, Suboption New Environment Option, Suboption Terminal Type
13	12.653625	192.168.1.10	192.168.1.20	TELNET	63	Suboption Negotiate About Window Size
14	12.695438	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=37 Ack=34 Win=64256 Len=0
15	12.695569	192.168.1.10	192.168.1.20	TELNET	70	Suboption New Environment Option, Suboption Terminal Type
16	12.696710	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=37 Ack=50 Win=64256 Len=0
17	12.704807	192.168.1.20	192.168.1.10	TELNET	69	Will Suppress Go Ahead, Do Echo, Do Linemode, Will Status, Do Remote Flow Control
18	12.705445	192.168.1.10	192.168.1.20	TELNET	57	Do Suppress Go Ahead
19	12.747605	192.168.1.10	192.168.1.20	TCP	60	23 → 54892 [ACK] Seq=52 Ack=53 Win=64256 Len=0
20	12.747798	192.168.1.10	192.168.1.20	TELNET	66	Will Echo, Won't Linemode, Don't Status, Won't Remote Flow Control
21	12.749380	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=52 Ack=65 Win=64256 Len=0
22	12.749380	192.168.1.20	192.168.1.10	TELNET	66	Don't Echo, Will Echo, Do Timing Mark, Do Binary Transmission
23	12.751103	192.168.1.10	192.168.1.20	TELNET	57	Won't Echo
24	12.753011	192.168.1.20	192.168.1.10	TELNET	140	86 bytes data
25	12.753293	192.168.1.10	192.168.1.20	TELNET	63	Do Echo, Won't Timing Mark, Will Binary Transmission
26	12.795411	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=150 Ack=77 Win=64256 Len=0
27	19.557444	192.168.1.10	192.168.1.20	TELNET	55	1 byte data
28	19.558710	192.168.1.20	192.168.1.10	TCP	60	23 → 54892 [ACK] Seq=150 Ack=78 Win=64256 Len=0
29	19.558710	192.168.1.20	192.168.1.10	TELNET	60	1 byte data
30	19.610613	192.168.1.10	192.168.1.20	TCP	54	54892 → 23 [ACK] Seq=78 Ack=151 Win=65384 Len=0
31	19.754212	192.168.1.10	192.168.1.20	TELNET	55	1 byte data
32	19.755904	192.168.1.20	192.168.1.10	TELNET	60	1 byte data
33	19.799293	192.168.1.10	192.168.1.20	TCP	54	54892 → 23 [ACK] Seq=79 Ack=152 Win=65384 Len=0
34	20.511753	192.168.1.10	192.168.1.20	TELNET	56	2 bytes data
35	20.513061	192.168.1.20	192.168.1.10	TELNET	60	4 bytes data
36	20.568212	192.168.1.10	192.168.1.20	TCP	54	54892 → 23 [ACK] Seq=81 Ack=156 Win=65380 Len=0
37	20.569438	192.168.1.20	192.168.1.10	TELNET	70	16 bytes data
38	20.613169	192.168.1.10	192.168.1.20	TCP	54	54892 → 23 [ACK] Seq=81 Ack=172 Win=65364 Len=0

FIGURE 11 – Capture Wireshark du trafic Telnet utilisant TCP

2.2.2 Identifiez les messages échangés entre le client Telnet et le serveur. Quels messages sont utilisés pour établir la connexion TCP initiale ?

Comme pour FTP, l'établissement de la connexion TCP utilise le **three-way handshake** :

1. [**SYN**] : Paquet #3 - Le client initie la connexion ;
2. [**SYN, ACK**] : Paquet #4 - Le serveur accepte ;
3. [**ACK**] : Paquet #5 - Le client confirme ;

Ensuite, on observe les messages spécifiques au protocole Telnet :

— Options de négociation Telnet (Will/Don't/Do) ;

— Authentification (demande de login et mot de passe) ;

— Transmission des commandes et des résultats ;

Les messages Telnet visibles incluent :

— "Will Authentication Option" ;

— "Will Encryption Option" ;

— "Do Terminal Type" ;

— "Don't Echo" ;

2.2.3 Quelles sont les valeurs des ports source et destination ? Quel est leur objectif ?

Dans la capture, on observe :

— **Port destination (serveur)** : **23** - Port Telnet standard (fixe)

— **Port source (client)** : **82740** - Port éphémère (variable)

Le fonctionnement est identique à FTP : le port serveur est fixe (23) tandis que le port client change à chaque connexion.

2.2.4 Si vous ouvrez une nouvelle connexion Telnet, les numéros de port changent-ils ?

Oui, le port source côté client change à chaque nouvelle connexion Telnet.

```
hathouti@Taha-inspiron-16:~$ sudo netstat -tulpn | grep :23
tcp6       0      0 :::23                   :::*                    LISTEN      82740/xinetd
hathouti@Taha-inspiron-16:~$ sudo netstat -tulpn | grep :23
tcp6       0      0 :::23                   :::*                    LISTEN      83275/xinetd
```

FIGURE 12 – Vérification avec netstat - Port Telnet (23) en écoute via xinetd

Le port 23 est géré par xinetd (PID 83275), qui crée une nouvelle instance pour chaque connexion entrante avec un port client différent.

2.2.5 Analysez les autres messages échangés, en particulier les messages TCP

Dans la capture Wireshark, on observe les échanges suivants :

1. Établissement TCP (paquets 3-5) :

— Paquet #3 : [SYN] - Initiation de la connexion ;

— Paquet #4 : [SYN, ACK] - Acceptation du serveur ;

— Paquet #5 : [ACK] - Confirmation du client ;

2. Négociation des options Telnet (paquets 6-22) :

— Paquet #6 : "Will Authentication Option, Will Encryption Option, Do Terminal Type, Do Terminal Speed..." ;

— Paquet #8 : "Don't Authentication Option" ;

— Paquet #10 : "Don't Encryption Option, Will Terminal Type, Will Negotiate About Window Size..." ;

— Paquet #12 : "Do Negotiate About Window Size, Suboption New Environment Option, Suboption Terminal Type" ;

— Paquet #15 : "Suboption New Environment Option, Suboption Terminal Type" ;

— Paquet #17 : "Will Suppress Go Ahead, Do Echo, Do Linemode..." ;

— Paquet #20 : "Will Echo, Won't Linemode, Don't Status, Won't Remote Flow Control" ;

— Paquet #22 : "Don't Echo, Won't Timing Mark, Do Binary Transmission" ;

3. Transmission de données (paquets 24+) :

— Paquet #24 : 86 bytes de données (écran de login) ;

— Paquet #27 : 1 byte data (caractère tapé) ;

— Paquet #29 : 1 byte data (caractère tapé) ;

— Paquet #31 : 1 byte data (caractère tapé) ;

— Paquet #34 : 2 bytes data ;

Observations importantes :

— Les messages TCP utilisent les flags [ACK], [PSH, ACK] *(malheuresuement pas pris en capture)* pour l'accusé de réception et la transmission ;

— **Problème majeur de sécurité :** Toutes ces données sont transmises **en clair (non chiffré)** ;

— Chaque caractère tapé génère un paquet individuel de 1-2 bytes ;

— Les numéros de séquence TCP augmentent progressivement (Seq=0, Seq=22, Seq=37, etc.) ;

Constat de sécurité :

— Bien que les identifiants ne soient pas directement visibles dans cette capture Wireshark, ils restent interceptables puisque le protocole ne chiffre pas les données ;

— Chaque caractère tapé génère un paquet visible ;

3 Capture et Analyse du Trafic du Protocole SSH

3.1 Processus de Capture

3.1.1 Démarrage du Service SSH

Sur la machine Ubuntu, le service SSH (OpenSSH) est généralement installé par défaut. Nous l'avons démarré avec :

```
sudo service ssh start
sudo service ssh status
```

```
hathouti@Taha-inspiron-16:~$ sudo netstat -tulpn | grep :22
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1/init
tcp6       0      0 :::22                   :::*                    LISTEN      1/init
```

FIGURE 13 – Vérification du port 22 (SSH) avec netstat - Service en écoute

Le port 22 est bien en écoute sur toutes les interfaces (0.0.0.0 :22 et : : :22).

3.1.2 À quelle couche du modèle OSI appartient le protocole SSH ? Quel est son numéro de port ?

— **Couche OSI :** SSH appartient à la **couche Application (couche 7)** du modèle OSI.

— **Port utilisé :** Port **22** (port standard pour SSH)

3.1.3 Connexion SSH depuis Windows

Depuis la machine Windows, nous avons établi une connexion SSH vers le serveur Ubuntu en utilisant PuTTY ou la commande SSH native :

```
ssh hathouti@192.168.1.20
```

3.2 Analyse du Trafic SSH

3.2.1 Quel protocole de la couche transport SSH utilise-t-il ?

SSH utilise le protocole **TCP (Transmission Control Protocol)** sur le port 22.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.20	192.168.1.255	UDP	86	57621 → 57621 Len=44
2	0.001098	192.168.1.20	192.168.1.255	UDP	86	57621 → 57621 Len=44
3	3.016715	192.168.1.20	192.168.1.255	UDP	86	57621 → 57621 Len=44
4	22.398791	192.168.1.20	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
5	22.488732	192.168.1.20	239.255.255.250	SSDP	167	M-SEARCH * HTTP/1.1
6	30.127620	192.168.1.10	192.168.1.20	TCP	66	55312 → 22 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=4 SACK_PERM
7	30.129832	192.168.1.20	192.168.1.10	TCP	66	22 → 55312 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
8	30.130066	192.168.1.10	192.168.1.20	TCP	54	55312 → 22 [ACK] Seq=1 Ack=1 Win=65532 Len=0
9	30.139011	192.168.1.10	192.168.1.20	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_for_Windows_9.5)
10	30.141209	192.168.1.20	192.168.1.10	TCP	60	22 → 55312 [ACK] Seq=1 Ack=34 Win=64256 Len=0
11	30.148085	192.168.1.20	192.168.1.10	SSHv2	97	Server: Protocol (SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.14)
12	30.151778	192.168.1.10	192.168.1.20	SSHv2	1486	Client: Key Exchange Init
13	30.157453	192.168.1.20	192.168.1.10	SSHv2	1174	Server: Key Exchange Init
14	30.162485	192.168.1.10	192.168.1.20	SSHv2	102	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
15	30.172931	192.168.1.20	192.168.1.10	SSHv2	546	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=284)
16	30.213501	192.168.1.10	192.168.1.20	TCP	54	55312 → 22 [ACK] Seq=1514 Ack=1656 Win=65532 Len=0
17	33.019183	192.168.1.20	192.168.1.255	UDP	86	57621 → 57621 Len=44
18	35.485108	BizlinkTechn_40:15:…	MicroStarINT_36:34:…	ARP	60	Who has 192.168.1.10? Tell 192.168.1.20
19	35.485199	MicroStarINT_36:34:…	BizlinkTechn_40:15:…	ARP	42	192.168.1.10 is at 04:7c:16:36:34:4a
20	45.781839	192.168.1.10	192.168.1.20	SSHv2	70	Client: New Keys
21	45.824655	192.168.1.20	192.168.1.10	TCP	60	22 → 55312 [ACK] Seq=1656 Ack=1530 Win=67200 Len=0
22	45.824702	192.168.1.10	192.168.1.20	SSHv2	98	Client: Encrypted packet (len=44)
23	45.826583	192.168.1.20	192.168.1.10	TCP	60	22 → 55312 [ACK] Seq=1656 Ack=1574 Win=67200 Len=0
24	45.826583	192.168.1.20	192.168.1.10	SSHv2	98	Server: Encrypted packet (len=44)
25	45.826900	192.168.1.10	192.168.1.20	SSHv2	122	Client: Encrypted packet (len=68)
26	45.831423	192.168.1.20	192.168.1.10	SSHv2	106	Server: Encrypted packet (len=52)
27	45.879504	192.168.1.10	192.168.1.20	TCP	54	55312 → 22 [ACK] Seq=1642 Ack=1752 Win=65436 Len=0
28	51.087453	192.168.1.10	192.168.1.20	SSHv2	202	Client: Encrypted packet (len=148)
29	51.129802	192.168.1.20	192.168.1.10	TCP	60	22 → 55312 [ACK] Seq=1752 Ack=1790 Win=70144 Len=0
30	51.164299	192.168.1.20	192.168.1.10	SSHv2	82	Server: Encrypted packet (len=28)
31	51.197079	192.168.1.10	192.168.1.20	SSHv2	166	Client: Encrypted packet (len=112)
32	51.199268	192.168.1.20	192.168.1.10	TCP	60	22 → 55312 [ACK] Seq=1780 Ack=1902 Win=70144 Len=0
33	51.240080	192.168.1.20	192.168.1.10	SSHv2	682	Server: Encrypted packet (len=628)
34	51.254685	192.168.1.10	192.168.1.20	SSHv2	634	Client: Encrypted packet (len=580)
35	51.256966	192.168.1.20	192.168.1.10	SSHv2	98	Server: Encrypted packet (len=44)
36	51.258320	192.168.1.10	192.168.1.20	SSHv2	190	Client: Encrypted packet (len=136)
37	51.260660	192.168.1.20	192.168.1.10	SSHv2	594	Server: Encrypted packet (len=540)
38	51.260660	192.168.1.20	192.168.1.10	SSHv2	162	Server: Encrypted packet (len=108)
39	51.260952	192.168.1.10	192.168.1.20	TCP	54	55312 → 22 [ACK] Seq=2618 Ack=3100 Win=64088 Len=0

FIGURE 14 – Capture Wireshark du trafic SSH - Connexion chiffrée

3.2.2 Identifiez les messages échangés entre le client SSH et le serveur. Quels messages sont utilisés pour établir la connexion TCP initiale ?

1. Établissement TCP (Three-Way Handshake) :

1. Paquet #6 : [SYN] - Le client (192.168.1.10 :55312) initie la connexion vers le serveur (192.168.1.20 :22) ;
2. Paquet #7 : [SYN, ACK] - Le serveur accepte la connexion ;
3. Paquet #8 : [ACK] - Le client confirme ;

2. Négociation SSH (Protocol SSHv2) :

— Paquet #9 : Client envoie "Protocol (SSH-2.0-OpenSSH_for_Windows_9.5)" ;
— Paquet #11 : Server répond "Protocol (SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.14)" ;
— Paquet #12-13 : Échange de clés (Key Exchange Init) ;
— Paquet #14 : "Elliptic Curve Diffie-Hellman Key Exchange Init" ;
— Paquet #15 : "Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys" ;

3. Authentification et Session Chiffrée :

— À partir du paquet #20 : Tous les paquets sont marqués "Encrypted packet" ;
— Le contenu (identifiants, commandes, résultats) est totalement chiffré ;

3.2.3 Quelles sont les valeurs des ports source et destination ? Quel est leur objectif ?

Dans la capture, on observe :

— **Port destination (serveur)** : **22** - Port SSH standard (fixe) ;
— **Port source (client)** : **55312** - Port éphémère (variable) ;

Le mécanisme est identique à FTP et Telnet.

```
hathouti@Taha-inspiron-16:~$ sudo netstat -tulpn | grep :22
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1/init
tcp6       0      0 :::22                   :::*                    LISTEN      1/init
```

FIGURE 15 – Observation netstat - Connexion SSH établie avec port client 55312

3.2.4 Si vous ouvrez une nouvelle connexion SSH, les numéros de port changent-ils ?

Oui, comme pour FTP et Telnet :

— Le **port serveur (22)** reste constant ;
— Le **port client** change à chaque nouvelle connexion ;

Cela permet de gérer plusieurs sessions SSH simultanées vers le même serveur.

3.2.5 Analysez les autres messages échangés, en particulier les messages TCP. Que remarquez-vous ?

Observation majeure : Contrairement à Telnet, le trafic SSH est **entièrement chiffré** après l'établissement de la connexion.

- **Phase initiale (claire)** :
  - Three-way handshake TCP ;
  - Négociation de version SSH ;
  - Échange de clés publiques ;
- **Phase chiffrée** :
  - Tous les paquets après l'échange de clés sont marqués "Encrypted packet" ;
  - Le contenu est illisible dans Wireshark ;
  - Les identifiants, commandes et données sont protégés ;
  - Taille variable des paquets chiffrés (44, 68, 148, 580 bytes...) ;
- **Avantages de SSH par rapport à Telnet** :
  - Confidentialité : Impossible d'intercepter les identifiants ;
  - Intégrité : Les données ne peuvent pas être modifiées en transit ;

Dans cette capture Wireshark, on voit clairement que tous les paquets après l'échange de clés (Paquet #15) sont marqués "Encrypted packet", rendant le contenu totalement illisible.

4 Conclusion

4.1 Synthèse des Observations

Ce TP nous a permis de comprendre l'importance de la confidentialité dans les communications réseau en analysant trois protocoles :

Critère	FTP	Telnet	SSH
Port	21 (commande) 20 (données)	23	22
Couche OSI	Application (7)	Application (7)	Application (7)
Transport	TCP	TCP	TCP
Chiffrement	Non	Non	Oui
Sécurité	Faible	Très faible	Élevée
Identifiants visibles	Oui	Oui	Non
Usage recommandé	Non	Non	Oui

TABLE 1 – Comparaison des trois protocoles analysés

4.2 Enseignements sur la Triade CIA

4.2.1 Confidentialité

- **FTP et Telnet** : Violation grave de la confidentialité
  - Les identifiants sont transmis en clair ;
  - Toutes les données sont interceptables avec Wireshark ;
- **SSH** : Protection de la confidentialité
  - Chiffrement de bout en bout ;
  - Impossibilité d'intercepter les données sensibles ;

4.2.2 Intégrité

- **FTP et Telnet** : Pas de protection contre les modifications ;
- **SSH** : Garantie l'intégrité ;

4.2.3 Disponibilité

Les trois protocoles utilisent TCP, qui garantit la fiabilité de la transmission, mais ne protègent pas contre les attaques DDoS qui peuvent affecter la disponibilité.