

Polytech Dijon FISA IOT 5A

Programmation pour l'embarquée - TP

Aspect réseau et sécurité

Auteur : CAMUS Pierre-Marie

Tuteur : BARD Patrick



2024-2025

Sommaire

1	Cor	npte Rendu	4
	1.1	Introduction	4
	1.2	Commandes	4
		1.2.1 Adresse IP	4
		1.2.2 Connexion SSH	4
		1.2.3 Transfert de fichier	4
		1.2.4 Programmation	5
		1.2.5 Analyse	5
		1.2.6 Commandes	6
		1.2.7 Programmation	6
		1.2.8 Analyse	8
	1.3	TD RUST	8
	1.4	Conclusion	10

Table des Figures

1.1	Paramètre réseau	4
1.2	Serveur.py	6
1.3	client.py	6
1.4	Wireshark réseau privé	6
1.5	Wireshark réseau privé2	7
1.6	Wireshark chiffré	8

1 Compte Rendu

1.1 Introduction

Le but de ce T.P. est de reviser les aspects réseau et sécurité avec des commandes de base

1.2 Commandes

1.2.1 Adresse IP

Avec la commande ip a ont obtient les paramètres des cartes réseau présent sur l'ordinateur.

ı ip a

```
IneMSL:-$ ip a
1. lo: <LODPBCK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127, 00:1/8 scope host lo
    valid.lft forever preferred.lft forever
    inet 10:255,255.50/2 brd 10:255.255.50/su scope global lo
    valid.lft forever preferred.lft forever
    inet 10:255,255.50/2 brd 10:255.255.50/su scope global lo
    valid.lft forever preferred.lft forever
    inet 6::1/120 scope host
    valid.lft forever preferred.lft forever
2: ethe: <a href="https://documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/documento.org/document
```

FIGURE 1.1 – Paramètre réseau

1.2.2 Connexion SSH

Pour l'utillisation de la commande SSH il faut stipuler un utilisateur et l'adresse ip du PC a atteindre (exemple : root et 192.168.0.111)

ssh root@192.168.0.111

1.2.3 Transfert de fichier

Pour l'utilisation de la commande SCP il faut stipuler l'utilisateur et l'adresse ip du PC plus le chemin ou on veut copier les fichiers

scp root@192.168.0.111:/usr/

1.2.4 Programmation

Création du code en python coté serveur pour afficher les infos envoyé par le client. Ne disposant pas de deux machines, ici on utilise le localhost avec le port 65432 qui est dit "Dynamique et éphémères".

```
import socket
   # Paramètres de connexion
   HOST = "127.0.0.1" # Localhost
   PORT = 65432
                         # Port d'écoute (au choix, >1024)
   # Création du socket serveur
   with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as serveur:
       serveur.bind((HOST, PORT))
       serveur.listen(1)
       print(f"Serveur en attente de connexion sur {HOST}:{PORT}...")
11
12
       conn, addr = serveur.accept()
13
       with conn:
14
           print(f"Connecté par {addr}")
15
           while True:
               data = conn.recv(1024) # Réception de données (max 1024 octets)
               if not data: # Si plus de données, on arrête
18
                   break
19
               print("Reçu :", data.decode("utf-8"))
20
21
```

Lancement du code dans un terminal

py serveur.py

A l'inverse Creation du code en python coté client pour envoyer les infos au serveur

```
import socket
  # Paramètres du serveur
  HOST = "127.0.0.1"
  PORT = 65432
   with socket.socket(socket.AF INET, socket.SOCK STREAM) as client:
       client.connect((HOST, PORT))
       print("Connecté au serveur. Tapez vos messages (q pour quitter) :")
       while True:
10
           msg = input(">>> ")
11
           if msg.lower() == "q": # Quitter
12
               break
           client.sendall(msg.encode("utf-8"))
14
15
16
```

1.2.5 Analyse

En utilisant la loopback wireshark ne voit pas le reseau lo.

```
Windows PowerShell X + V
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités s

PS C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau> py .\serveur.py
Serveur en attente de connexion sur 127.0.0.1:65432...
Connecté par ('127.0.0.1', 51313)
Recu : hello world
```

FIGURE 1.2 – Serveur.py

FIGURE 1.3 – client.py

Du coup changement de technique. On crée avec E.HELLES un réseau privé en utilisant la partage de connexion d'un meme telephone portable. Je fais le serveur et lui le client. Sur les images ci dessous on voit les echanges coté terminal et coté Wireshark. Avec le message en clair

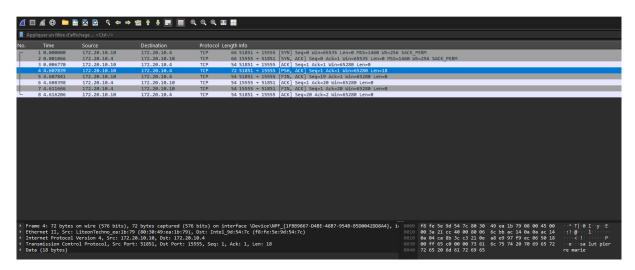


FIGURE 1.4 – Wireshark réseau privé

1.2.6 Commandes

Commande openssl pour la génération de clé RSA classique privé 2048bits.

- openssl genpkey -algorithm RSA -out private.pem -pkeyopt rsa_keygen_bits:2048

 Commande openssl pour la extraite la clé RSA publique.
- openssl rsa -pubout -in private.pem -out public.pem

1.2.7 Programmation

On garde la même configuration, je transfère ma clé publique issue de ma clé privée à E.HELLE pour qu'il puisse chiffrer les données et que je sois le seul à déchiffrer le message.

- import socket
- 2 from cryptography.hazmat.primitives.asymmetric import padding

```
OUTPUT
                    DEBUG CONSOLE
                                    TERMINAL
PS C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau> py .\serveur.py
    data = conn.recv(1024) # Réception de données (max 1024 octets)
ConnectionResetError: [WinError 10054] Une connexion existante a dû être fermée par l'hôte distant
PS C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau> py .\serveur.py
Serveur en attente de connexion sur 172.20.10.4:15555...
Connecté par ('172.20.10.10', 51836)
Traceback (most recent call last):
  File "C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau\serveur.py", line 17, in <module>
    data = conn.recv(1024) # Réception de données (max 1024 octets)
ConnectionResetError: [WinError 10054] Une connexion existante a dû être fermée par l'hôte distant
PS C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau> py .\serveur.py
Serveur en attente de connexion sur 172.20.10.4:15555...
Connecté par ('172.20.10.10', 51841)
Reçu : bjr
PS C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau> py .\serveur.py
Serveur en attente de connexion sur 172.20.10.4:15555...
Connecté par ('172.20.10.10', 51851)
Reçu : salut pierre marie
PS C:\Users\eltou\Documents\ProgEmbarque\TD3_Revision_Reseau>
```

FIGURE 1.5 – Wireshark réseau privé2

```
from cryptography.hazmat.primitives import serialization, hashes
   HOST = "172.20.10.4"
   PORT = 15555
   # Charger la clé privée depuis un fichier PEM
8
   with open("privatePM.pem", "rb") as f:
9
       private key = serialization.load pem private key(f.read(), password=None)
10
       print("Clé privée chargée")
12
   with socket.socket(socket.AF INET, socket.SOCK STREAM) as serveur:
13
       serveur.bind((HOST, PORT))
       serveur.listen(1)
15
       print(f"Serveur en attente de connexion sur {HOST}:{PORT}...")
16
17
       conn, addr = serveur.accept()
       with conn:
19
           print(f"Connecté par {addr}")
20
           while True:
21
                data = conn.recv(256) # Limité à un bloc RSA
22
                if not data:
23
                    break
25
                try:
26
                    # Déchiffrement RSA
27
                    message = private_key.decrypt(
28
                        data,
29
                        padding.OAEP(
30
                             mgf=padding.MGF1(algorithm=hashes.SHA256()),
                             algorithm=hashes.SHA256(),
                             label=None
33
                        )
34
                    )
35
```

```
print("Message déchiffré :", message.decode("utf-8"))
except Exception as e:
print("Erreur de déchiffrement :", e)
```

1.2.8 Analyse

Sur l'image ci-dessous issue de Wireshark, on voit la trame passer mais ce coup-ci le message est chiffré et beaucoup plus long qu'avant

ip.addr == 172.20.10.4							
No. Time	Source	Destination	Protocol Le	enath Info			
1 0.000000 2 0.001520 3 0.00552698 7 4.423090 8 4.501310 9 4.614565 10 4.658774 11 7.649418 12 7.651212 13 7.696776	34.120.52.64 172.20.10.4 34.120.52.64 172.20.10.4 162.159.136.234 162.159.136.234 172.20.10.4 34.120.52.64 172.20.10.4 34.120.52.64	172.20.10.4 34.120.52.64 172.20.10.4 162.159.136.234 172.20.10.4 162.159.136.234 172.20.10.4 162.159.136.234 172.20.10.4 34.120.52.64	TLSv1.2 TLSv1.2 TCP TLSv1.2 TCP TLSv1.2 TCP TLSv1.2 TLSv1.2 TLSv1.2	54 443 → 60553 [ACK] Seq=1 Ack=119 Win=16 Len=0 88 Application Data 54 60553 → 443 [ACK] Seq=119 Ack=35 Win=253 Len=0			
71 12.199467	172.20.10.10	172.20.10.4	TCP	310 52189 + 15555 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=256			
L 72 12.242617 74 12.852113 75 12.930005 76 15.261653 77 15.308928	172, 29, 19, 4 172, 29, 119, 4 185, 125, 190, 56 162, 159, 136, 234 172, 29, 10, 4	172, 28, 18, 18 185, 125, 190, 56 172, 28, 18, 4 172, 28, 18, 4 162, 159, 136, 234	TCP NTP TLSv1.2 TCP	54 1555 + 52189 [ACK] Seq-1 Ack-257 Win-251 Len-0 90 HTP Version 4, Citent 90 HTP Version 4, server 96 Application Data 54 606553 + 443 [ACK] Seq-119 Ack-77 Win-253 Len-0			
 Ethernet II, Src: Internet Protocol Transmission Contr Data (256 bytes) 	LiteonTechno_ea:1b:79 Version 4, Src: 172.2 rol Protocol, Src Port	9 (80:30:49:ea:1b:79), 20.10.10, Dst: 172.20 t: 52189, Dst Port: 1	, Dst: Intel .10.4 5555, Seq: 1,	on interface \Device\MPF_[1889867-D48E-4887-9548-850004200000 94:54172 (f8:fe:5e:94:5417c) Ack: 1, Len: 256	0010 01 28 21 f8 40 00 80 0 0020 0a 04 cb dd 3c c3 fc 2 0030 00 ff d2 e9 00 00 84 4 0040 86 a0 f2 59 a4 2b e7 0	5 6b a1 ac 14 0a 0a ac 14 d 26 1b 58 28 80 73 50 18 c d9 86 c6 45 30 66 a7 c2 l 00 81 38 02 af 38 f5 8d	.^T 01.y.E (!@k <&X(sP L.E0f Y++ & &
[Length: 256]					9856 14 c0 5e 38 e 2 a6 df d 9970 27 e5 15 36 94 55 c 9 e e 9880 38 35 dc ee 59 9b c17 e 9880 38 45 3b 14 4 e 7 d 54 9c 6 9a 9b 9 9a 9b 6 9a 9b 9 9a 9	7 ca 45 de 77 5 fe e9 6d 7a e9 6d 17a e9 6d 10 48 an ea 58 6.8 dd ca e2 cc ce 7a e9 72 85 52 52 52 6d	** ** ** ** ** ** ** ** ** ** ** ** **

FIGURE 1.6 – Wireshark chiffré

Lien Github

1.3 TD RUST

Pour approximer le chiffre PI, il faut créer un carré de 2 unité et a l'intérieur un cercle unitaire de 1 en utilisant la methode monte carlo. On génère un grand nombre de points aléatoires dans un carré de côté 2 (de -1 à 1).

On compte combien tombent dans le cercle de rayon $1 \times X \times X + Y \times Y \leq 1$.

Comme l'aire du cercle vaut $\pi * r * r = \pi$ et l'aire du carré vaut 4, la proportion est :

$$\pi \approx 4 \cdot \frac{N_{\text{cercle}}}{N_{\text{total}}}.$$

```
import random

def approx_pi(nb_points: int) -> float:
    inside_circle = 0

for _ in range(nb_points):
    x = random.uniform(-1, 1)
    y = random.uniform(-1, 1)

if x**2 + y**2 <= 1:
    inside_circle += 1</pre>
```

```
return 4 * inside_circle / nb_points

return 4 * inside_circle / nb_points

**Exemple avec 1 million de points

return 4 * inside_circle / nb_points

Lie Github
```

1.4 Conclusion

Ce TP a permis de se refamilliariser avec les commandes de bases et l'aspect sécurité des messages