TP 7 – Sécurité

I) Socket sécurisé

I) Certificat X509

1)

ca.pem : CA = Certificat d'Autorité → confirme l'identité d'un serveur ou client et permet de les lier à des clés privées.

ca.key : délivre la clé privée à l'entité donnée par ca.pem .

server.pem : confirme l'identité du propriétaire du serveur, et permet de le lier à une clé privée.

server.key : délivre la clé privée au propriétaire du serveur donner par server.pem .

2)

— version du certificat :

```
Version: 3
```

— numéro de série de certificat :

```
Serial Number (hex): 1eb74c7337400d26d85ec1bc592a468308bb9700
```

— nom du propriétaire du certificat :

```
Subject: CN=127.0.0.1
Subject Alternative Name (not critical):
        DNSname: 127.0.0.1
        IPAddress: 127.0.0.1
```

— nom de l'autorité de certification :

```
Issuer: CN=CA
```

— date d'expiration du certificat :

```
Validity:
        Not Before: Tue Dec 06 20:57:26 UTC 2022
        Not After: Sun Aug 31 20:57:32 UTC 2025
```

— key id :

```
7f01fab60f74003147ea9b23ff4d341f0e1b7d4b
```

— signature :

```
Signature:
        72:6a:58:c7:15:a4:11:cb:42:d0:2d:b2:fd:89:c0:67
        0a:1d:1e:a5:2d:ce:ec:d3:32:c6:19:3a:f5:c2:8d:02
        c4:45:10:34:67:de:56:4b:c6:bb:8e:f1:6a:c9:3a:92
        f3:db:c3:e7:53:99:ee:a4:84:45:06:fc:b1:a1:48:9f
        de:7a:27:7d:30:37:f3:ef:08:23:50:fb:34:00:fe:0f
        18:41:bb:5e:3b:36:99:0d:a5:d9:09:b5:24:c0:e3:e1
        29:37:b1:4b:64:b5:2d:5e:16:3d:66:e7:3e:ff:67:ef
        56:64:b8:12:0e:c8:1f:cb:1f:1c:ee:42:c2:9f:9d:1d
        55:30:48:68:3d:a8:f0:76:1b:45:a5:c3:3a:f3:a2:5a
        7c:1e:13:a6:f7:e8:7e:50:24:82:97:34:84:d6:9d:13
        33:37:c4:55:ff:48:26:e7:12:07:a9:e2:72:98:69:66
        a0:8a:11:34:c2:6f:9a:ea:34:68:bd:d0:bf:0d:61:ba
        30:16:25:2c:48:fd:bc:06:c8:a7:25:4c:b5:e0:6b:6e
        d9:c8:b6:a7:64:c4:3c:33:b5:3f:28:7d:4a:2c:41:c1
        4b:aa:a9:b4:77:fc:75:a1:2c:2a:80:14:26:b6:9e:4c
        61:c4:84:9b:2c:10:98:31:50:ea:8c:d1:15:4e:9d:66
        20:9b:6e:db:8c:41:42:a0:5d:a9:42:13:a3:fc:69:22
        6f:9e:8e:ed:18:2b:34:e8:42:b4:c1:63:52:bc:79:b3
        48:87:37:cf:55:08:f0:66:bc:c5:32:60:75:32:f5:08
        6c:29:b6:e5:ee:86:b1:5e:26:ed:fc:6e:8b:2f:b4:a6
        b1:d3:da:f9:ee:81:68:eb:c0:94:79:80:33:cd:bb:bb
        ba:a2:21:eb:9f:f8:2e:a7:30:62:9f:84:6f:21:67:0d
        35:a2:a7:ba:7b:d4:a8:62:40:12:e4:c7:ac:35:9b:a8
        d6:e0:27:c3:2f:1f:3a:6d:d5:87:fb:a1:ed:5a:ba:80
```

— fingerprint :

```
Fingerprint:
        sha1:4bf706d9d65c8a57a449d20c261e00ef6af8a57d
        sha256:3d492034098eb162696663ac76a48b006c6437537c718a9332d6e87e2896e353
```

— valeur de la clé publique :

```
pin-sha256:JBraOMQgir19WvT+aLykTIqrzKDyOCCHsW1wygs56uU=
```

II) Mise en œuvre du certificat

1) tout est conforme, aucune erreur est affichée.

2) L’erreur est du au fait que le certificat n’est pas reconnu par le navigateur.

```
SHA-1      4B:F7:06:D9:D6:5C:8A:57:A4:49:D2:0C:26:1E:00:EF:6A:F8:A5:7D
```

des données sensibles pourraient être récupérées.
Après avoir rajouté une exception, le site est accessible.

III) Programmation Socket SSL en Python

```python
#!/usr/bin/python3
import socket
import ssl

HOST = '127.0.0.1'
PORT = 7777
BUFSIZE = 1024

# echo server
def echo(sc):
    while True:
        data = sc.recv(BUFSIZE)
        if data == b'' or data == b'\n':
            break
        print(data.decode())
        sc.sendall(data)


# main program
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((HOST, PORT))
s.listen()
context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.load_cert_chain("server.pem","server.key")
sslsock = context.wrap_socket(s, server_side=True)

while True:
    sc, addr = sslsock.accept()
    echo(sc)
    sc.close()

s.close()

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

kbalavoine@sheeana:~/Bureau/espaces/travail/FAC/L2 Info/Reseaux/TP7$ python3 server.py
Hello World!
```

```python
#!/usr/bin/python3
import socket
import ssl

HOST = "127.0.0.1"
PORT = 7777
BUFSIZE = 1024

# main program
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)
context.load_verify_locations("ca.pem","ca.key")
s.connect((HOST, PORT))
sslsock = context.wrap_socket(s, server_hostname=HOST)
msg = b"Hello World!"
sslsock.sendall(msg)
answer = sslsock.recv(BUFSIZE)
print(answer.decode())
sslsock.close()

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

kbalavoine@sheeana:~/Bureau/espaces/travail/FAC/L2 Info/Reseaux/TP7$ python3 client.py
Hello World!
kbalavoine@sheeana:~/Bureau/espaces/travail/FAC/L2 Info/Reseaux/TP7$
```