

# W3L2 - Consegna

ESERCIZIO PROGRAMMAZIONE PER HACKER – PYTHON



# Cos'è una backdoor e perché è una vulnerabilità?

- Una backdoor è punto di accesso nascosto deliberatamente inserito in un sistema informatico o un'applicazione.
- Consente a un utente non autorizzato di ottenere l'accesso al sistema senza passare attraverso i normali controlli di sicurezza. Le backdoor possono essere utilizzate per diversi scopi, come la manutenzione di sistemi o l'accesso di emergenza.
- Le backdoor vengono spesso sono associate a scopi malevoli, come il hacking e l'accesso non autorizzato ai dati o ai sistemi. Gli hacker possono sfruttare le backdoor per compromettere la sicurezza di un sistema informatico o di una rete. Pertanto, la scoperta e la chiusura delle backdoor sono essenziali per garantire la sicurezza informatica.



Questo codice è un semplice client Python che stabilisce una connessione a un server specificato tramite un indirizzo IP e una porta. Una volta stabilita la connessione, il client offre un menu di opzioni all'utente, consentendo di eseguire le seguenti azioni:

- Chiudere la connessione (0): L'utente può digitare "0" per chiudere la connessione al server e terminare il programma.
- Ottenere informazioni di sistema (1): L'utente può digitare "1" per richiedere al server informazioni sul sistema. Il server invierà le informazioni di sistema al client e verranno stampate a schermo.
- Elenca il contenuto di una directory (2): L'utente può digitare "2" per richiedere al server di elencare il contenuto di una directory specifica. L'utente dovrà inserire il percorso della directory, il client invierà questa richiesta al server, riceverà la lista dei file e delle directory nella directory specificata e la stamperà a schermo.
- Il client utilizza il modulo socket per gestire la connessione con il server. La connessione viene stabilita all'inizio del programma e chiusa quando l'utente sceglie l'opzione "0" dal menu. Le altre opzioni inviano richieste al server e ricevono dati in risposta, che vengono poi elaborati e stampati a schermo.

```
1 import socket
2
3 SRV_ADDR = input("Type the server IP address: ")
4 SRV_PORT = int(input("Type the server port: "))
5
6 def print_menu():
7     print("""\n\n0) Close the connection
8 1) Get system info
9 2) List directory contents""")
10
11 my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12 my_sock.connect((SRV_ADDR, SRV_PORT))
13
14 print("Connection established")
15 print_menu()
16
17 while 1:
18     message = input("\n-Select an option: ")
19
20     if(message == "0"):
21         my_sock.sendall(message.encode())
22         my_sock.close()
23         break
24
25     elif(message == "1"):
26         my_sock.sendall(message.encode())
27         data = my_sock.recv(1024)
28         if not data: break
29         print(data.decode('utf-8'))
30
31     elif(message == "2"):
32         path = input("Insert the path: ")
33         my_sock.sendall(message.encode())
34         my_sock.sendall(path.encode())
35         data = my_sock.recv(1024)
36         data = data.decode('utf-8').split(",")
37         print("*"*40)
38         for x in data:
39             print(x)
40         print("*"*40)
41
```

Questo codice crea un server socket che ascolta su un indirizzo IP e una porta specificati (192.168.32.100 e 1234) per le connessioni in ingresso. Una volta stabilita una connessione con un client, il server accetta richieste dal client e fornisce risposte in base ai comandi inviati dal client. Ecco cosa fa in modo sintetico:

- Il server crea un socket e lo associa a un indirizzo IP e una porta specificati.
- Il server inizia ad ascoltare per le connessioni in ingresso.
- Quando una connessione viene stabilita con successo, il server accetta la connessione e ottiene l'indirizzo del client.
- Il server entra in un loop infinito in cui attende comandi dal client.
- Se il client invia "1", il server risponde inviando informazioni sulla piattaforma e l'architettura del sistema operativo su cui è in esecuzione.
- Se il client invia "2", il server attende ulteriori dati dal client (presumibilmente un percorso di directory), elenca i file nella directory specificata e invia l'elenco al client.
- Se il client invia "0", il server chiude la connessione corrente con il client e si mette in ascolto per una nuova connessione.

Questo codice rappresenta un server di base che può essere utilizzato per eseguire azioni specifiche basate sui comandi inviati dal client, come ottenere informazioni sul sistema o elencare i file in una directory specifica.

```
1 import socket, platform, os
2
3 SRV_ADDR = "192.168.32.100"
4 SRV_PORT = 1234
5
6 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 s.bind((SRV_ADDR, SRV_PORT))
8 s.listen(1)
9 connection, address = s.accept()
10
11 print ("client connected: ", address)
12
13 while 1:
14     try:
15         data = connection.recv(1024)
16         except:continue
17
18         if(data.decode('utf-8') == '1'):
19             tosend = platform.platform() + " " + platform.machine()
20             connection.sendall(tosend.encode())
21         elif(data.decode('utf-8') == '2'):
22             data = connection.recv(1024)
23             try:
24                 filelist = os.listdir(data.decode('utf-8'))
25                 tosend = ""
26                 for x in filelist:
27                     tosend += "," + x
28             except:
29                 tosend = "Wrong path"
30             connection.sendall(tosend.encode())
31         elif(data.decode('utf-8') == '0'):
32             connection.close()
33             connection, address = s.accept()
34
```

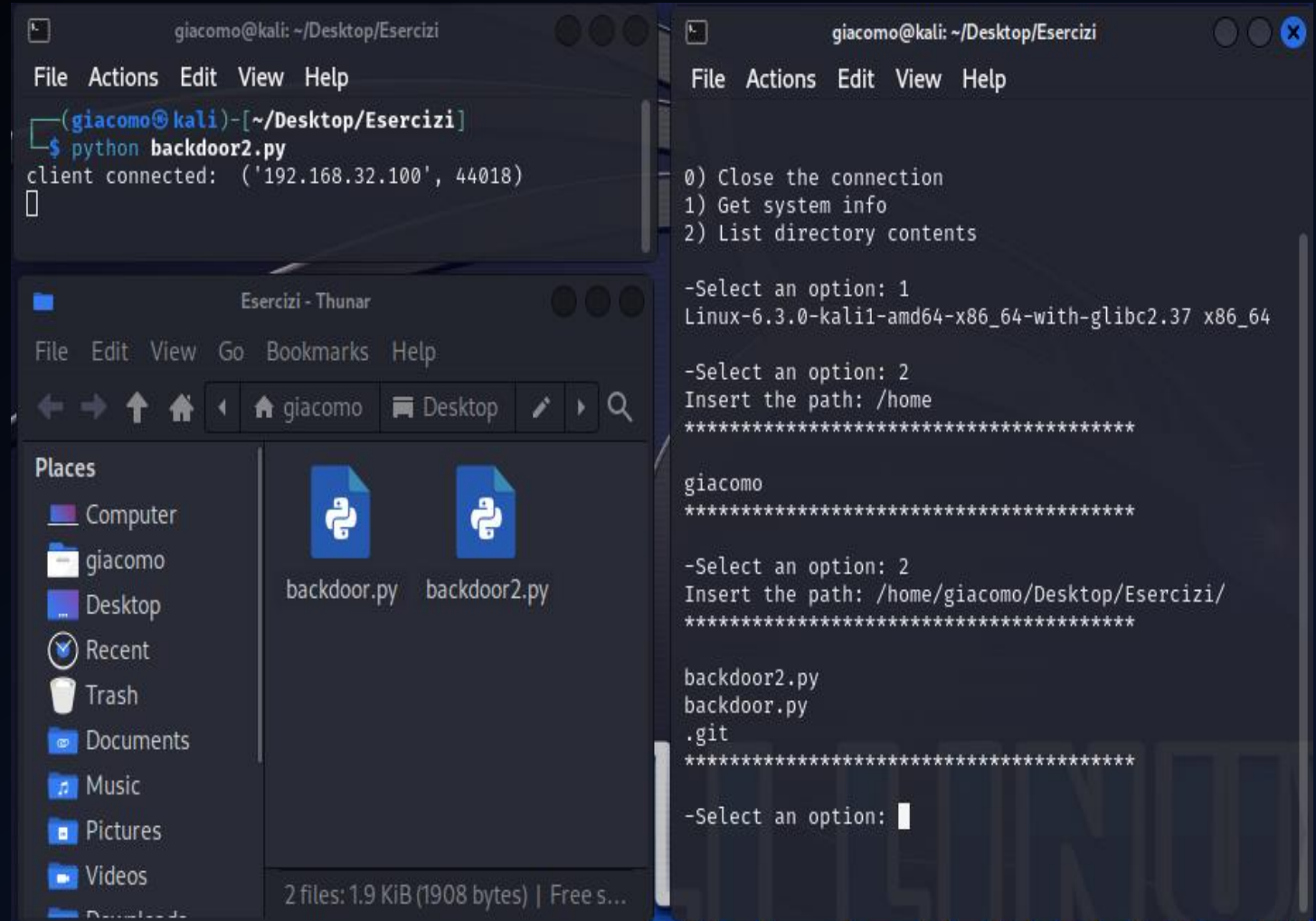


## Testiamo il codice

In questo caso il client e il server sono lo stesso dispositivo e condividono lo stesso IP, essendo la simulazione svolta in macchina virtuale.

Possiamo osservare che il socket osserva la connessione in ingresso.

Abbiamo inoltre richiesto di elencare il contenuto della depository "/home/giacomo/Desktop/Esercizi/" e possiamo osservare i due file in essa contenuti.



The screenshot displays a Kali Linux desktop environment. On the left, a terminal window titled 'giacomo@kali: ~/Desktop/Esercizi' shows the execution of the command `python backdoor2.py`. The output indicates a client connection from '192.168.32.100' on port 44018. Below the terminal, a file manager window titled 'Esercizi - Thunar' shows the contents of the directory '/home/giacomo/Desktop/Esercizi/'. The file manager displays two Python files: 'backdoor.py' and 'backdoor2.py'. On the right, another terminal window titled 'giacomo@kali: ~/Desktop/Esercizi' shows the output of the backdoor script. It prompts the user to select an option (0) to close the connection, (1) to get system info, or (2) to list directory contents. The user selects option 1, and the script outputs system information: 'Linux-6.3.0-kali1-amd64-x86\_64-with-glibc2.37 x86\_64'. The user then selects option 2, and the script prompts for a path. The user enters '/home', and the script lists the contents of the directory: 'giacomo', 'backdoor2.py', 'backdoor.py', and '.git'.

```
giacomo@kali: ~/Desktop/Esercizi
File Actions Edit View Help
(giacomo@kali)-[~/Desktop/Esercizi]
$ python backdoor2.py
client connected: ('192.168.32.100', 44018)

Esercizi - Thunar
File Edit View Go Bookmarks Help
giacomo Desktop
Places
Computer
giacomo
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
2 files: 1.9 KiB (1908 bytes) | Free s...

giacomo@kali: ~/Desktop/Esercizi
File Actions Edit View Help
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
*****

giacomo
*****

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

backdoor2.py
backdoor.py
.git
*****

-Select an option: 
```