



Esercizio S3L2 cos' è una  
backdoor e perché è pericolosa



## Cos'è:

Una backdoor è un programma o una funzionalità di un sistema informatico che consente a un utente non autorizzato di accedere a un sistema o a una rete. Le backdoor possono essere utilizzate per scopi legittimi, come consentire agli amministratori di sistema di accedere a un sistema da remoto per eseguire operazioni di manutenzione, ma possono anche essere utilizzate per scopi dannosi, come rubare dati o installare malware.



# Perché sono pericolose?

Le backdoor sono pericolose perché possono consentire a un utente non autorizzato di accedere a un sistema o a una rete senza essere rilevato. Questo può consentire all'utente malintenzionato di eseguire operazioni dannose, come:

Rubare dati, come informazioni personali, finanziarie o commerciali.

Installare malware, come virus, trojan o ransomware.

Controllare il sistema o la rete, ad esempio per eseguire attacchi DDoS.



# Come proteggersi?

Per proteggersi dalle backdoor, è importante mantenere aggiornati i sistemi operativi e le applicazioni software. È inoltre importante utilizzare un firewall e un antivirus per rilevare e bloccare le intrusioni.

Ecco alcuni consigli per proteggere i sistemi informatici dalle backdoor:

Installando gli aggiornamenti di sicurezza non appena sono disponibili.

Utilizzando un firewall per bloccare l'accesso non autorizzato da Internet.

Utilizzando un antivirus per rilevare e bloccare il malware.

Eseguendo regolarmente la scansione del sistema alla ricerca di malware.

Educando gli utenti sui rischi delle backdoor e su come proteggersi.

## Spiegazione primo codice fornito

```
1 import socket, platform, os
2
3 SRV_ADDR = "192.168.32.100"
4 SRV_PORT = 80
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
```



Riga 1: Importa le librerie socket, platform e os. Queste librerie forniscono le funzionalità necessarie per creare un server web.

Riga 2: Dichiarare le costanti SRV\_ADDR e SRV\_PORT. Queste costanti specificano l'indirizzo IP e la porta del server.

Riga 3: Crea un oggetto socket. Un socket è un'entità software che consente a due programmi di comunicare tra loro.

Riga 4: Associa il socket all'indirizzo IP e alla porta specificati.

Riga 5: Imposta il socket in modalità "ascolto". Ciò significa che il socket è pronto a ricevere connessioni da altri programmi.

Riga 6: Accetta una connessione da un altro programma.

Riga 7: Stampa l'indirizzo IP del client che si è connesso.

Riga 8: Entra in un ciclo infinito.

Riga 9: Riceve dati dal client.


Riga 10: Prova a decodificare i dati ricevuti in formato UTF-8.

Riga 11: Se la decodifica ha avuto successo, procede con il resto dell'elaborazione. Altrimenti, continua al ciclo successivo.

Riga 12: Controlla il valore dei dati ricevuti.

Riga 13: Se il valore è 1, invia al client le informazioni sul sistema operativo e sulla macchina.

Riga 14: Se il valore è 2, invia al client l'elenco dei file presenti nella directory specificata dai dati ricevuti.



Riga 15: Prova a decodificare i dati ricevuti in formato UTF-8.

Riga 16: Se la decodifica ha avuto successo, procede con il resto dell'elaborazione. Altrimenti, invia al client il messaggio "Wrong path".

Riga 17: Crea una lista dei file presenti nella directory specificata.

Riga 18: Per ogni file nella lista, invia al client il nome del file.

Riga 19: Invia al client il messaggio "Wrong path".


Riga 20: Chiude la connessione con il client.

Riga 21: Ritorna al ciclo successivo.

## spiegazione secondo codice fornito

```
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     if(message == "0"):
20         my_sock.sendall(message.encode())
21         my_sock.close()
22         break
23
24     elif(message == "1"):
25         my_sock.sendall(message.encode())
26         data = my_sock.recv(1024)
27         if not data: break
28         print(data.decode('utf-8'))
29
30     elif(message == "2"):
31         path = input("Insert the path: ")
32         my_sock.sendall(message.encode())
33         my_sock.sendall(path.encode())
34         data = my_sock.recv(1024)
35         data = data.decode('utf-8').split(",")
36         print("*"*40)
37         for x in data:
38             print(x)
39         print("*"*40)
```





Riga 1: Importa i moduli `socket`, `platform` e `os`. Il modulo `socket` fornisce le funzionalità per creare e utilizzare socket, che sono un tipo di connettore di rete. Il modulo `platform` fornisce informazioni sul sistema operativo e sull'hardware in uso. Il modulo `os` fornisce funzionalità per interagire con il sistema operativo.

Riga 2: Definizione delle variabili `SRV_ADDRESS` e `SRV_PORT` specifica l'indirizzo IP del server a cui si desidera connettersi. `SRV_PORT` specifica la porta del server a cui si desidera connettersi.

Riga 3: Creazione di un oggetto socket. L'oggetto socket viene utilizzato per comunicare con il server.

Riga 4: Associazione dell'oggetto socket all'indirizzo IP e alla porta specificati.

Riga 5: Ascolto di connessioni in entrata. Il server è ora in ascolto di connessioni in entrata da parte dei client.

Riga 6: Accettazione di una connessione in entrata. L'oggetto `connection` rappresenta la connessione con il client. L'oggetto `address` contiene l'indirizzo IP e la porta del client.

Riga 7: Stampa dell'indirizzo IP e della porta del client.


Riga 8: Ciclo infinito. Il ciclo viene eseguito fino a quando il server viene arrestato.

Riga 9: Ricezione dei dati dal client. I dati ricevuti dal client vengono memorizzati nell'oggetto `data`.

Riga 10: Decodifica dei dati ricevuti. I dati ricevuti dal client sono codificati in formato UTF-8. L'oggetto `data` viene decodificato in formato testo.

Riga 11: Controllo del valore dei dati ricevuti. Il valore dei dati ricevuti viene utilizzato per determinare cosa fare.

Riga 12: Caso `data == 1`. Se il valore dei dati ricevuti è uguale a "1", il server invia al client le informazioni sul sistema operativo e sull'hardware in uso.



Riga 13: Creazione della stringa da inviare al client. La stringa contiene le informazioni sul sistema operativo e sull'hardware in uso.

Riga 14: Codifica della stringa da inviare al client. La stringa viene codificata in formato UTF-8.

Riga 15: Invio della stringa al client. La stringa viene inviata al client.

Riga 16: Caso `data == "2"`. Se il valore dei dati ricevuti è uguale a "2", il server invia al client l'elenco dei file in una determinata directory.

Riga 17: Ricezione dei dati dal client. I dati ricevuti dal client vengono memorizzati nell'oggetto `data`.

Riga 18: Decodifica dei dati ricevuti. I dati ricevuti dal client sono codificati in formato UTF-8. L'oggetto `data` viene decodificato in formato testo.

Riga 19: Controllo del valore dei dati ricevuti. Il valore dei dati ricevuti viene utilizzato per determinare se la directory specificata esiste.


Riga 20: Caso `data == "wrong path"`. Se la directory specificata non esiste, il server invia al client la stringa "Wrong path".

Riga 21: Caso `data != "wrong path"`. Se la directory specificata esiste, il server invia al client l'elenco dei file nella directory.

Riga 22: Creazione della stringa da inviare al client. La stringa contiene l'elenco dei file nella directory.

Riga 23: Codifica della stringa da inviare al client. La stringa viene codificata in formato UTF-8.

Riga 24: Invio della stringa al client. La stringa viene inviata al client.



Riga 25: Caso `data == "0"`. Se il valore dei dati ricevuti è uguale a "0", il server chiude la connessione con il client.

Riga 26: Chiusura della connessione con il client.

Riga 27: Accettazione di una nuova connessione in entrata. Il server è ora in ascolto di connessioni in entrata da parte di nuovi client.