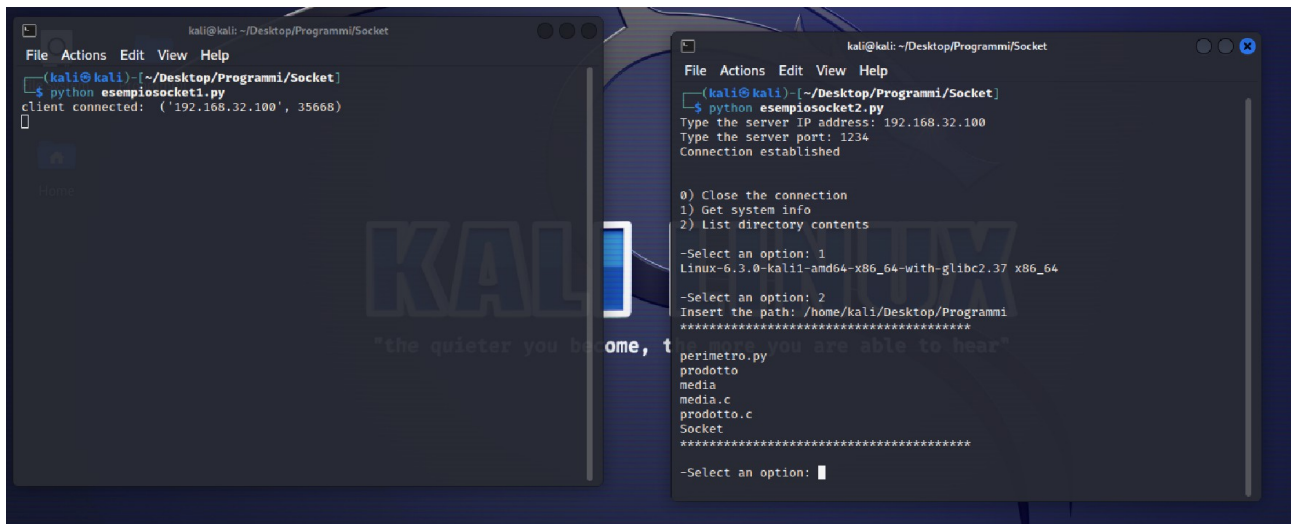


S3L2 - Pratica

Una backdoor è un modo per ottenere un accesso con diritti di amministratore su un sistema che non preveda una richiesta di autenticazione ed autorizzazione. Può essere una vulnerabilità del sistema stesso o può essere creata intenzionalmente dai programmatori per bypassare le procedure di autenticazione ed ottenere un accesso privilegiato

I due programmi di oggi servono a simulare una connessione client-server. Il primo codice mostrato rappresenta il codice lato server, mentre il secondo quello lato client. Nel primo codice ho inserito l'ip della macchina virtuale Kali, nel mio caso 192.168.32.100 lasciando 1234 come porta:



```
kali@kali: ~/Desktop/Programmi/Socket
File Actions Edit View Help
(kali@kali) ~/Desktop/Programmi/Socket
$ python esempiosocket1.py
client connected: ('192.168.32.100', 35668)

kali@kali: ~/Desktop/Programmi/Socket
File Actions Edit View Help
(kali@kali) ~/Desktop/Programmi/Socket
$ python esempiosocket2.py
Type the server IP address: 192.168.32.100
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/kali/Desktop/Programmi
*****
perimetro.py
prodotto
media
media.c
prodotto.c
Socket
*****

-Select an option: 1
```

Nell'esempio in figura vediamo come il programma 1, a sinistra, accetti la connessione del client una volta avviato il programma 2 da un'altra finestra di terminale, informandoci sul numero di porta sorgente del client. A destra vediamo come il programma 2 proponga una scelta tra:

- 0-Terminare la connessione
- 1-Ottenere info sul sistema
- 2-Listare il contenuto di una cartella

Scegliendo 1 ottengo info sulla versione di Linux installata sul sistema, mentre scegliendo 2 ottengo l'elenco dei file presenti nella cartella di cui ho inserito il path, nel mio esempio /home/kali/Desktop/Programmi

Spiegazione codice:

Programma1 – Server:

- 1 – Importo i moduli per gestire i socket e le comunicazioni di rete, l'accesso alle info di sistema (platform) e al sistema operativo per il path (os)
- 2 – Assegnazione delle variabili SRV-ADDR e SRV-PORT, rispettivamente l'indirizzo ip e la porta dove rimarrà in ascolto il server
- 3 - creazione del socket s, tramite AF_INET specifichiamo che useremo ipv4 e SOCK_STREAM che il flusso sarà TCP
- 4 – Associa il socket creato al punto 3 alle variabili ip e porta dichiarate al punto 2 tramite s.bind((SRV_ADDR, SRV_PORT))
- 5- Inizio dell'ascolto tramite s.listen(1), l'argomento 1 indica il numero massimo di connessioni che

il server accetterà

6 – Accettazione di una connessione con `connection, address = s.accept()`, e successiva stampa a schermo di “client connected:” seguito dal suo indirizzo e porta

7 – Ciclo `While(1)` che serve ad entrare in un loop infinito in attesa delle opzioni scelte dal client, che come da codice del programma 2 potranno essere 0,1 o 2. All'interno vediamo come accetti i dati dal client con un buffer di 1024 byte, con l'istruzione `data = connection.recv(1024)`

8 – Da qui inizia la gestione delle richieste del client.

- Se la scelta è 1 invia i dati `platform.platform()` e `platform.machine()` contenenti le info di sistema del server
- Se la scelta è 2 immagazzina il path inserito dall'utente nella variabile `filelist` con l'istruzione `filelist = os.listdir(data.decode('utf-8'))` ed invia la lista dei file presenti nel path specificato
- Se la scelta è 0 richiama la funzione `connection.close()` per chiudere la connessione tra client e server, tornando in ascolto con la funzione già vista `connection, address = s.accept()`

Programma 2 – Client

1- Dopo l'import del già visto metodo `socket`, chiede all'utente di inserire indirizzo ip e porta del server a cui si vuole connettere tramite `SRV_ADDR = input("Type the server IP address: ")` e `SRV_PORT = int(input("Type the server port: "))`

2 – Definisce una funzione chiamata `print_menu()` dove stampa a schermo le 3 scelte presenti in figura sopra

3 – Crea un socket per il client chiamato `mysock` tramite l'istruzione `my_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)` e cerca una connessione verso il server tramite l'istruzione `my_sock.connect((SRV_ADDR, SRV_PORT))`

4- Viene richiamata la funzione `print_menu()`

5 – Si entra in un ciclo `While(1)` come visto in precedenza chiedendo una scelta all'utente tramite l'istruzione `message = input("\n-Select an option: ")`

- Se la scelta è 0 viene inviata al server con `my_sock.sendall(message.encode())`, chiusa la connessione con `my_sock.close()` e si esce dal ciclo con `break`
- Se la scelta è 1 viene inviata la scelta al server come visto sopra, immagazzinata la risposta in una variabile chiamata `data` con `data = my_sock.recv(1024)` e stampata a schermo in formato `utf-8`. Se non vengono ricevuti dati si esce dal ciclo con `break`
- Se la scelta è 2 viene immagazzinato il path inserito dall'utente in una variabile `path` con `path = input("Insert the path: ")`, vengono immagazzinati come i prima di dati ricevuti con `data = my_sock.recv(1024)`, viene divisa in elementi separati da virgola con `data = data.decode('utf-8').split(",")` e successivamente tramite un ciclo `for` stampa i dati ricevuti a schermo, facendo variare una variabile `x` all'interno della variabile `data` appena ricevuta.