S3-L2 backdoor

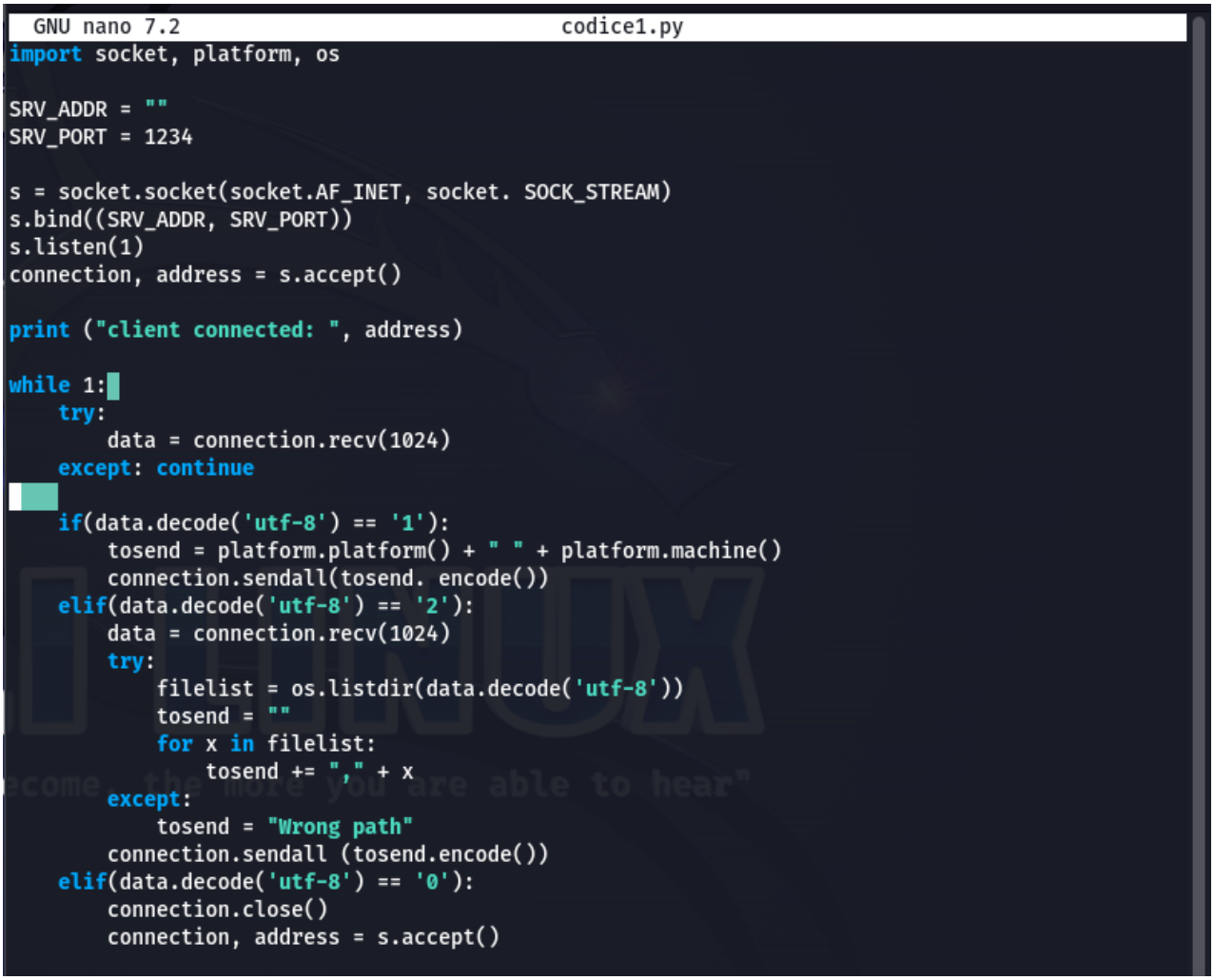
Traccia

Spiegare cos' è una backdoor e perchè è pericolosa.
Spiegare i codici qui sotto dicendo cosa fanno e qual è la differenza tra i due. Opzionale (consigliato) testare praticamente il codice.

Cos' è una backdoor e perché è pericolosa:

Una backdoor (o porta di servizio) in ambito informatico consente un accesso non autorizzato a un sistema o a dati.
Utilizzata dai programmatori, la backdoor permette un ingresso rapido e facilitato ai sistemi in caso di lecita manutenzione.
È pericolosa perché può essere utilizzata da malintenzionati per accedere a sistemi sensibili, rubare dati, installare malware o causare altri danni, senza che l'utente legittimo ne sia a conoscenza.

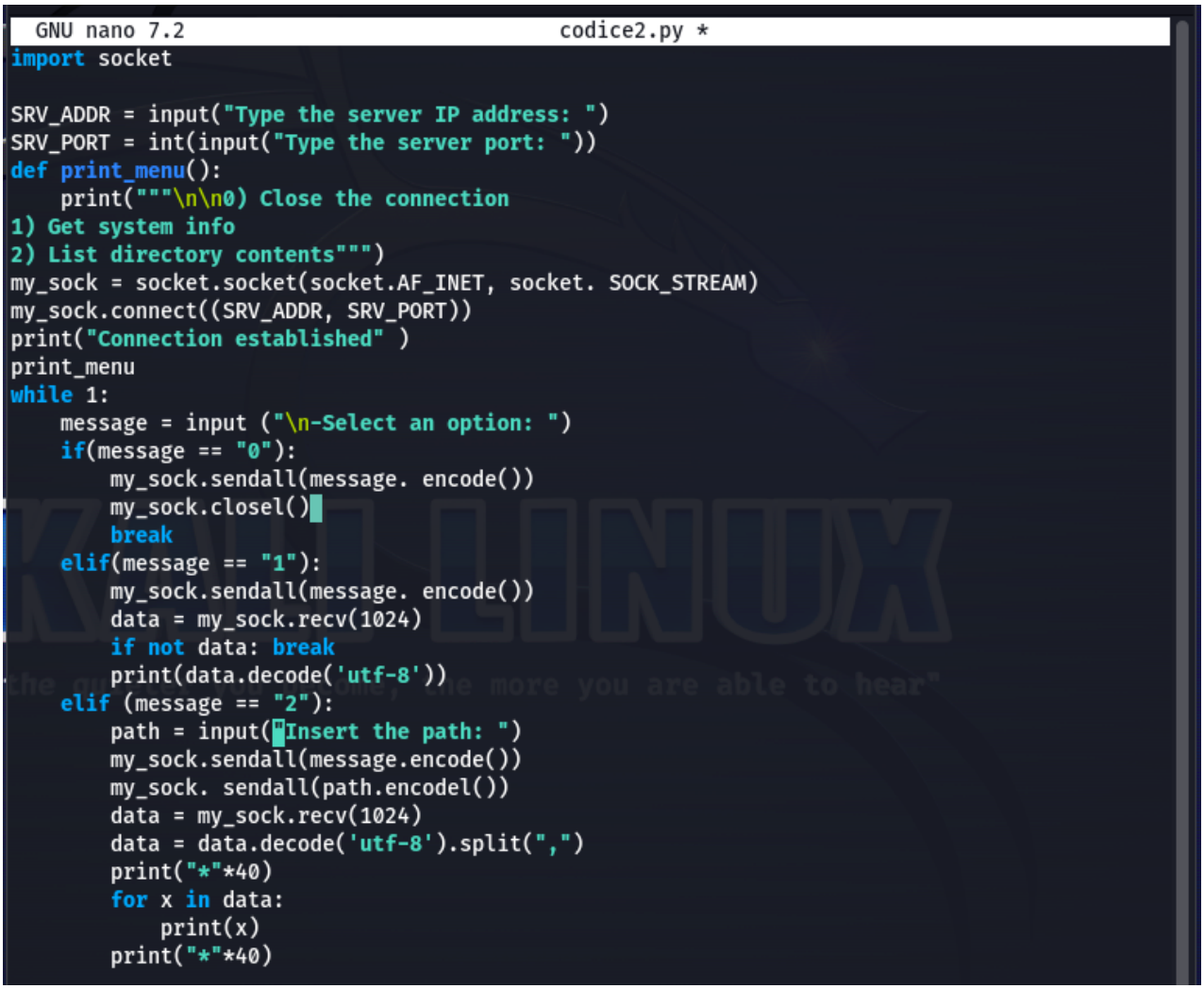
Primo codice:



Primo codice: **SERVER BACKDOOR**

1. Importazione delle Librerie:
   a. socket: Utilizzata per creare e gestire connessioni di rete tramite il protocollo TCP/IP. Fondamentale per ascoltare, accettare e gestire le connessioni in entrata.
   b. platform: Fornisce informazioni sulla piattaforma su cui viene eseguito il codice, come il sistema operativo e l'architettura hardware.
   c. os: Offre funzioni per interagire con il sistema operativo, come l'elencare i file in una directory.
2. Configurazione del Server:
   a. SRV_ADDR e SRV_PORT: Definiscono rispettivamente l'indirizzo IP e la porta su cui il server ascolterà le connessioni in entrata. In questo caso, l'indirizzo IP è vuoto, il che significa che il server è configurato per ascoltare le connessioni su tutte le interfacce di rete disponibili. Ai fini dell'esercizio inserisco qui l'IP della mia macchina virtuale Kali.
   b. socket.socket(): Crea un oggetto socket specificando il tipo di indirizzo (AF_INET per IPv4) e il tipo di socket (SOCK_STREAM per TCP).
   c. s.bind(): Associa il socket all'indirizzo IP e alla porta specificati.
   d. s.listen(1): Mette il server in ascolto per le connessioni in entrata. L'argomento 1 indica che può accettare una sola connessione in coda prima di rifiutare le connessioni successive.
3. Gestione delle Connessioni:
   a. accept(): Blocca l'esecuzione fino a quando un client non si connette. Una volta stabilita una connessione, restituisce un nuovo socket e l'indirizzo del client connesso.
   b. print("client connected: ", address): Stampa l'indirizzo del client connesso.
4. Ciclo di Gestione delle Richieste:
   a. Il ciclo while 1: è un loop infinito che attende e gestisce le richieste del client.
   b. connection.recv(1024): Tenta di ricevere dati dal client. Se il client si disconnette o si verifica un errore, il ciclo continua senza interrompersi.
   c. Gestione dei Comandi:
      - if '1': Invia al client informazioni sulla piattaforma e sull'architettura hardware.
      - elif '2': Riceve un percorso di directory dal client e invia un elenco dei file in quella directory, o un messaggio di errore se il percorso è errato.
      - elif '0': Chiude la connessione corrente e aspetta una nuova connessione.
5. Fine del Ciclo:
   a. Il ciclo infinito termina solo se il processo del server viene interrotto manualmente.

Secondo codice:



```
 GNU nano 7.2                                codice2.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.closel()
        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.encodel())
        data = my_sock.recv(1024)
        data = data.decode('utf-8').split(",")
        print("*"*40)
        for x in data:
            print(x)
        print("*"*40)
```

Secondo codice: **CLIENT BACKDOOR**

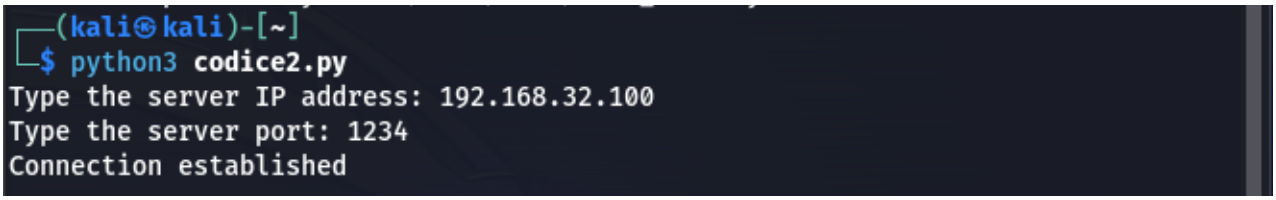
1. **Importazione della Libreria socket**:
   Utilizzata per stabilire una connessione TCP/IP con il server backdoor.
2. **Configurazione della Connessione**:
   SRV_ADDR e **SRV_PORT**: Richiedono all'utente di inserire l'indirizzo IP e la porta del server backdoor.
   my_sock.connect(): Stabilisce una connessione TCP con il server all'indirizzo e alla porta specificati.
3. **Interfaccia Utente e Menu**:
   Fornisce un'interfaccia a riga di comando con opzioni per l'utente, per interagire con il server backdoor.
4. **Ciclo di Invio dei Comandi**:
   - Un loop infinito che permette all'utente di inviare comandi al server.
   - **input()**: Richiede all'utente di selezionare un'opzione o di fornire input aggiuntivi (come il percorso della directory).
   - Gestione dei Comandi:
     if '0': Chiude la connessione e termina l'esecuzione del client.
     elif '1' e **elif '2'**: Inviano rispettivamente i comandi per ottenere informazioni di sistema e l'elenco dei file in una directory specificata.
5. **Ricezione e Visualizzazione dei Dati**:
   Riceve i dati dal server e li visualizza all'utente.

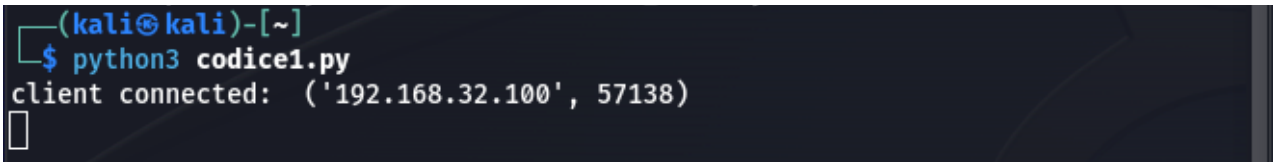
Differenze tra i due codici

- **Ruoli**: Il primo codice funge da server, in ascolto e risposta ai comandi, mentre il secondo è un client che inizia le azioni, inviando comandi e richieste al server.
- **Interazione**: Nel server, la maggior parte dell'interazione è automatizzata e reattiva ai comandi del client. Nel client, l'interazione è guidata dall'utente attraverso un'interfaccia a riga di comando.
- **Scopo**: Il server fornisce l'accesso e il controllo remoto del sistema su cui è eseguito, mentre il client è il mezzo attraverso cui un utente può esercitare questo controllo.

Test pratico del codice su Kali

Modifico il primo codice inserendo l'ip di Kali, 192.168.32.100. Lascio 1234 come porta ed eseguo il programma.
Eseguo su un secondo terminale il secondo codice stabilendo correttamente la connessione:





Sul mio secondo terminale, lato client, testo le opzioni "1" (per visualizzare il sistema operativo) e "2" (per visualizzare le cartelle/elenco di cartelle a seconda del path inserito).