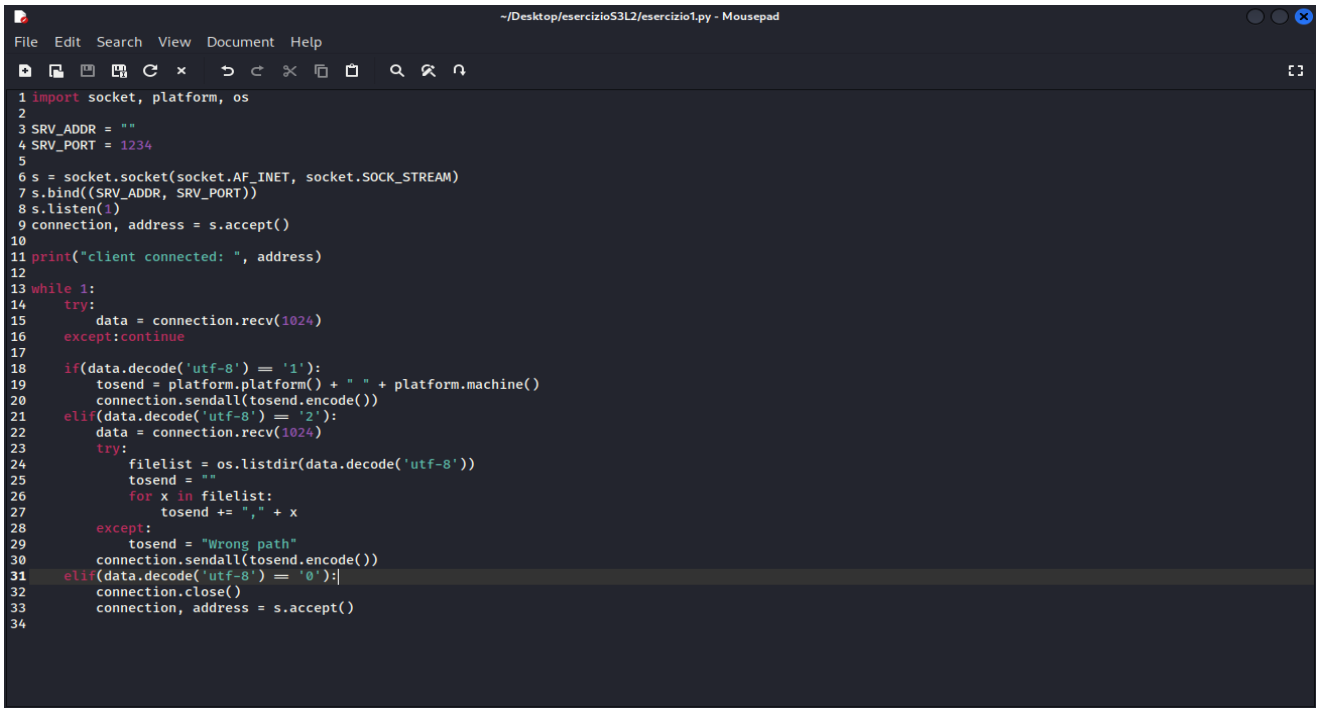
ESERCIZIO S3L2



Spiegazione del primo codice:

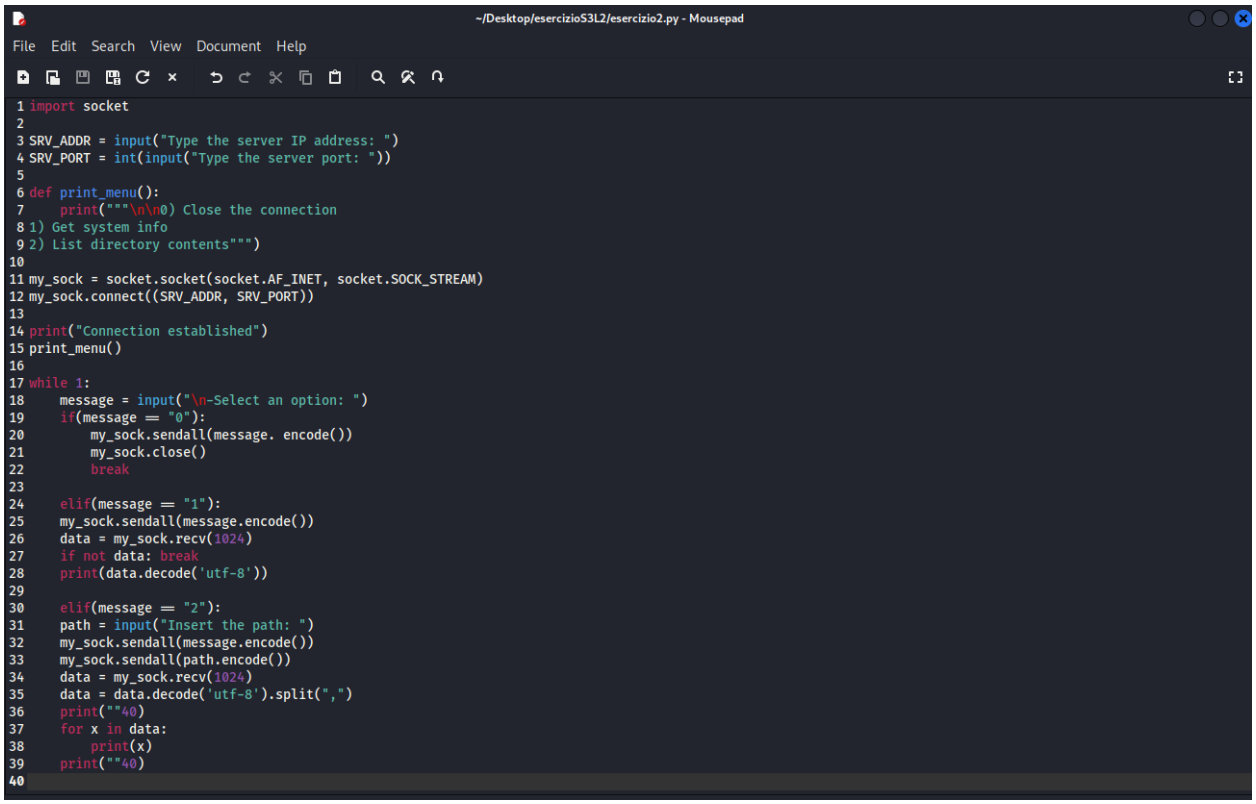
Il codice fornisce un esempio di un server socket in Python (ed e una backdoor). Il server ascolta su una porta specifica per connessioni in entrata da un client. Una volta stabilita una connessione, il server riceve comandi dal client attraverso la connessione e fornisce risposte in base a tali comandi. Le funzionalità principali includono:

1. Se il client invia il comando "1", il server risponde con informazioni sulla piattaforma e la macchina su cui è in esecuzione.
2. Se il client invia il comando "2", il server attende di ricevere un percorso e restituisce la lista dei file presenti in quella directory.
3. Se il client invia il comando "0", il server chiude la connessione corrente e si mette in attesa di una nuova connessione.

Spiegazione dettagliata:

- 1. **Inizializzazione del server:**
- - Viene creata una socket (punto finale per la comunicazione) usando `socket.socket()`.
- - La socket viene associata a un indirizzo e una porta specifici utilizzando `bind()`.
- - Il server entra in modalità di ascolto tramite `listen(1)`, consentendo di accettare una connessione alla volta.
-
- 2. **Accettazione di connessioni in entrata:**
- - Il server entra in uno stato di attesa tramite `accept()`, bloccandosi fino a quando non viene ricevuta una connessione in entrata da un client.
- - Una volta stabilita la connessione, viene accettata e si ottengono i dettagli di connessione come l'indirizzo IP e la porta del client.
-
- 3. **Comunicazione con il client:**
- - Il server entra in un loop infinito (`while 1`) per mantenere la connessione attiva e gestire i comandi inviati dal client.
- - Utilizzando `recv(1024)`, il server riceve dati dal client.
- - In base al comando ricevuto (`"1"`, `"2"`, o `"0"`), il server risponde con informazioni sulla piattaforma e la macchina, fornisce la lista dei file in una directory specifica o chiude la connessione corrente, rispettivamente.
-
- 4. **Chiusura della connessione:**
- - Se il client invia il comando `"0"`, il server chiude la connessione corrente attraverso `connection.close()`.
- - Successivamente, il server ritorna in modalità di ascolto per accettare potenziali nuove connessioni.

Questo codice rappresenta un esempio di interazione basica tra un server e un client, ma la mancanza di controlli adeguati sugli input del client lo rende vulnerabile a potenziali attacchi, rendendolo inadatto per un ambiente di produzione.



Spiegazione secondo codice:

Questo codice è un esempio di un client socket in Python che si connette a un server remoto attraverso una socket. Il client consente all'utente di interagire con il server attraverso un menu a selezione multipla.

Spiegazione dettagliata:

1. Inizializzazione delle variabili: L'utente inserisce l'indirizzo IP del server (SRV_ADDR) e la porta del server (SRV_PORT) tramite l'input.
2. Definizione della funzione print_menu(): Questa funzione stampa un menu a schermo che mostra le opzioni disponibili per l'utente.
3. Creazione della socket e connessione al server: Viene creata una socket (my_sock) utilizzando socket.socket(). La socket si connette al server remoto specificato attraverso l'indirizzo IP e la porta forniti dall'utente.

4. Visualizzazione del menu e loop principale: Dopo la connessione riuscita, viene stampato un messaggio di conferma e il menu viene visualizzato utilizzando print_menu(). Entra in un loop infinito (while 1) che consente all'utente di inviare comandi al server.
5. Selezione dell'opzione e interazione con il server: L'utente inserisce un numero corrispondente all'opzione desiderata dal menu. Se l'opzione è "0", il client invia al server il comando per chiudere la connessione e poi termina il loop. Se l'opzione è "1", il client invia al server il comando per ottenere informazioni di sistema e riceve e stampa la risposta del server. Se l'opzione è "2", il client invia al server il comando per elencare il contenuto di una directory specifica, inserendo il percorso e stampando la lista ricevuta dal server.

Spiegazione backdoor:

Una backdoor è una via segreta e intenzionalmente creata per bypassare normali procedure di autenticazione o per ottenere accesso non autorizzato a un sistema, applicazione o rete. Le backdoor possono essere implementate da sviluppatori, amministratori di sistema o utenti malevoli per diversi scopi, inclusi l'accesso segreto, il monitoraggio non autorizzato o l'esecuzione di azioni dannose sul sistema ospite.

Le backdoor sono pericolose per diverse ragioni:

1. Accesso non autorizzato: Le backdoor consentono l'accesso a un sistema senza la necessità di autenticazione normale. Ciò significa che chiunque conosca l'esistenza della backdoor e il metodo per utilizzarla può ottenere l'accesso, bypassando le misure di sicurezza standard.
2. Persistenza: Le backdoor spesso sono progettate per rimanere nascoste e persistere nel sistema nel tempo. Anche se vengono rilevate e rimosse, possono essere reintegrate nel sistema in modo da garantire un accesso continuato.
3. Attacchi silenziosi: Gli attaccanti possono utilizzare le backdoor per eseguire azioni dannose in modo discreto e silenzioso, senza che gli utenti o gli amministratori del sistema ne siano consapevoli. Ciò può facilitare furti di dati, monitoraggio non autorizzato o danni al sistema.
4. Difficili da rilevare: Le backdoor possono essere progettate per nascondersi dai controlli di sicurezza e dalle soluzioni antivirus. Questo le rende difficili da individuare e mitigare.
5. Abuso di privilegi: Le backdoor spesso operano con privilegi elevati, consentendo agli attaccanti di eseguire operazioni con ampi poteri, come modificare configurazioni di sistema, installare malware o alterare dati.

Per queste ragioni, la scoperta e la rimozione tempestiva delle backdoor sono fondamentali per mantenere la sicurezza dei sistemi informatici. La prevenzione, la vigilanza e l'implementazione di misure di sicurezza robuste sono essenziali per mitigare il rischio di backdoor e altri tipi di minacce alla sicurezza informatica.

Differenza tra i due codici:

Il primo fa da server e da backdoor (apre la porta all interno della macchina server), il secondo fa da client e puo connettersi al server tramite la backdoor inviando dei comandi.