

Title: TCP Port Scan

Name: Dikshya Kafle

Stu No: 2018380039

Class No: 10101801

Deadline: Dec 13th 2020

School of Computer Science and Engineering

Subject: Discover the TCP service opened by the remote host.

- Network ports are the contact endpoints for a computer that is linked to the Internet. When a service listens on a port, a client application may receive data, process it and communicate a response.
- Port scanning is part of a penetration test's first step which helps you to identify all available network entry points on a target device. For TCP and UDP ports, the port scanning methods are different, which is why we have dedicated tools for each one.

Purpose:

Through discovering the TCP service opened by the remote host, understand the working Principle of C/S communication mode and port scanning.

Requirement:

- (1) Run as command: DOS> scanPort remote_ip.
- (2) Output: TCP service port number opened by the remote host.

Implementation Principle:

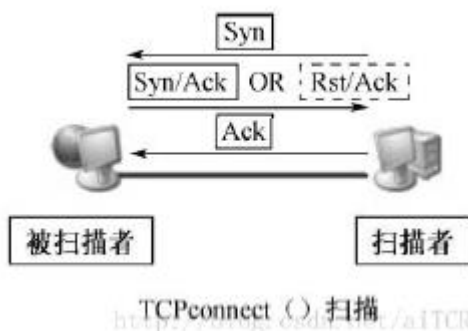
-Common methods: -TCP connect, TCP SYN, TCP FIN;

-Method used in this experiment: The OS provides the connect () system call to establish a connection with a port of the remote host. If the port of the remote host is in listening, the connect () connection is successful; otherwise, the port is closed;

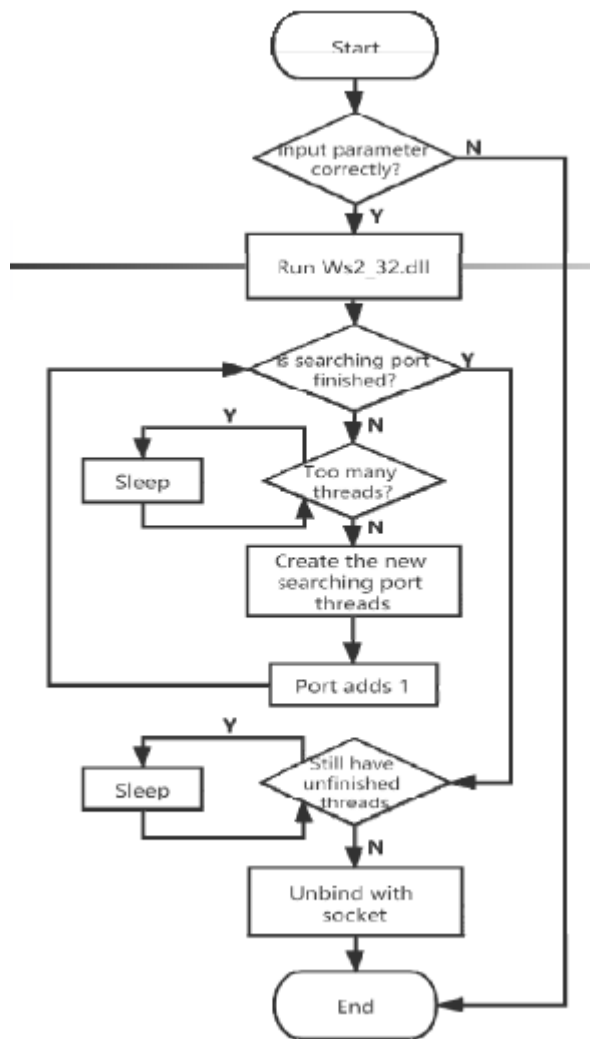
Advantage: No permission required. Any user can use the system call.

TCP Connect() Scan

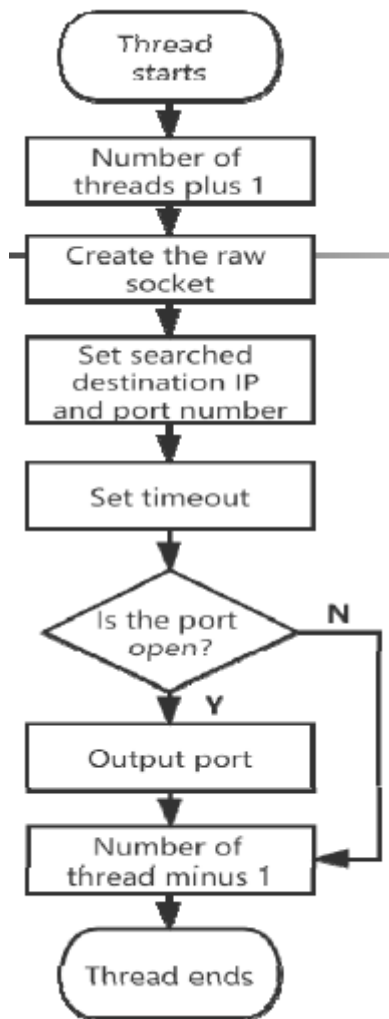
This is the most basic TCP scan, which use the connect() function provided by the system to connect to the target port and try to establish a complete three-way handshake process with the remote host. Therefore, this scanning method is also called "full scan". If the target port is in the listening state, connect () returns successfully, otherwise -1 is returned, indicating that the port is not accessible.



[Main program flowchart:](#)

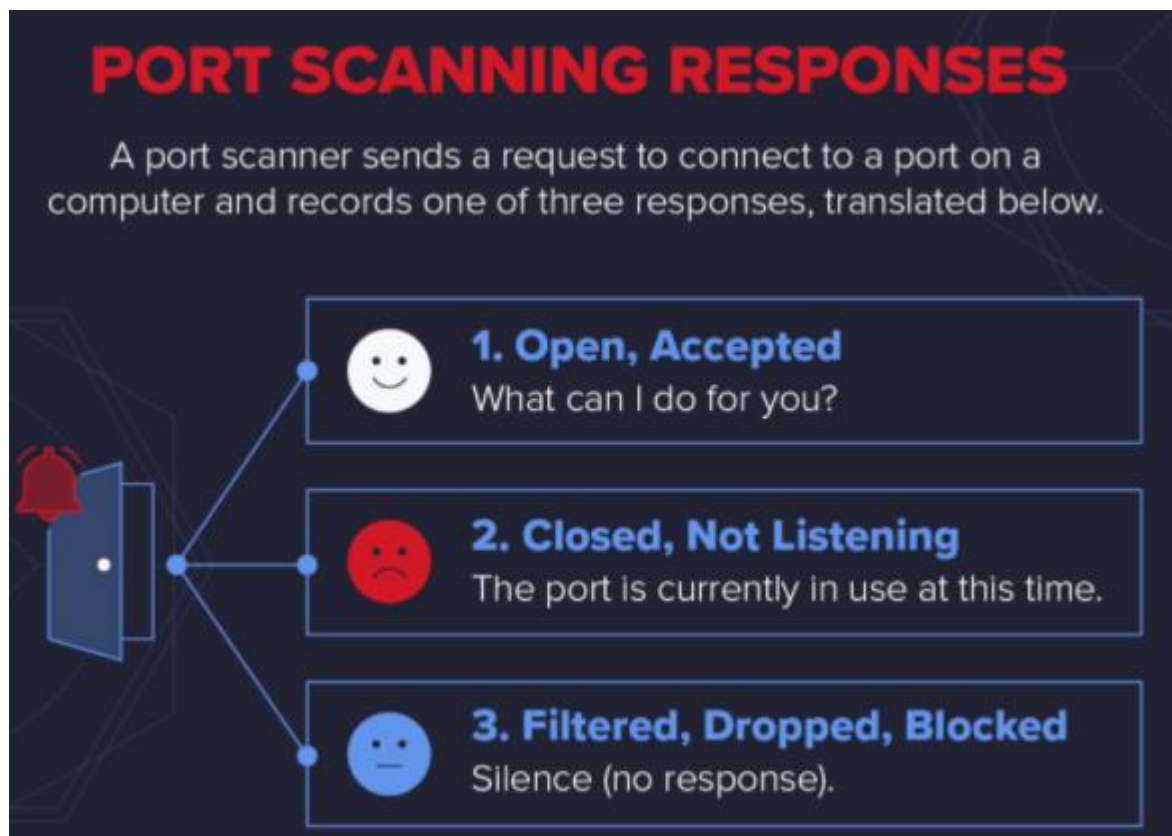


Sub-thread flowchart:

**Port Scan:**

A port scanner is a computer program that tests for one of three possible statuses, open, closed, or filtered, for network ports.

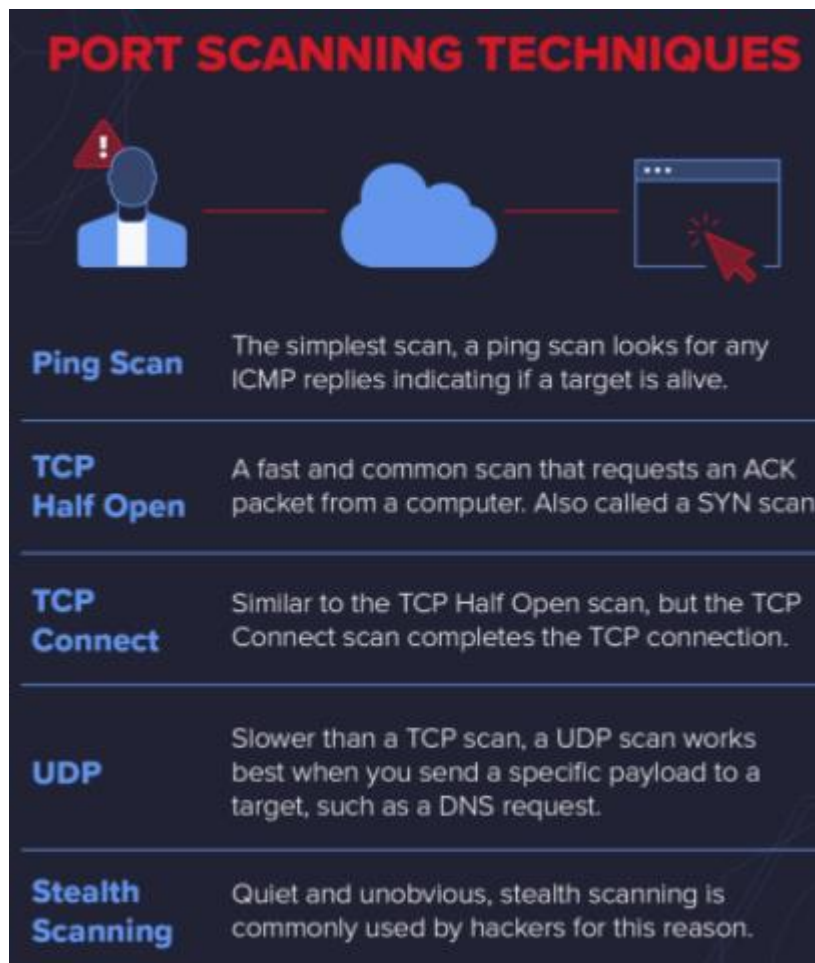
In diagnosing network and communication problems, port scanners are useful resources. Port scanners, however, are used by attackers to detect potential access points for intrusion and to determine what kinds of computers, such as firewalls, proxy servers or VPN servers, you are running on the network.



A network request to connect to a particular TCP or UDP port on a computer is submitted by a port scanner and the response is registered.

So what a port scanner does is send a network data packet to a port to verify its current state. If you decided to check if your web server was running properly, you would check the server's port 80 status to make sure that it was open and listening.

The status helps network engineers diagnose network problems or connectivity problems with the application, or helps attackers find potential ports to use for network infiltration.



TCP Connect:

Basically, this port scanning technique is the same as the TCP Half-Open scan, except the port scanner completes the TCP connection instead of leaving the target dangling.

It's not a method as common as the half-open TCP. First, per check, you have to send one more packet, which increases the amount of noise on the network you make. Second, as you complete the link to the target, you might activate an alarm that the half-open scan wouldn't do.

Target systems are more likely to log a complete TCP connection, and similarly, intrusion detection systems (IDS) are more likely to cause alarms from the same host on multiple TCP connections.

The benefit of the TCP connect scan is that to run the Half-open scan, a user doesn't need the same level of privileges as they do. The communications protocols any user requires to connect to other systems are used by TCP connect scans.

Port Scan:

A port scan is a technique for evaluating which ports are available on a network. Port scanning is similar to knocking on doors to see if anyone is home, since ports on a device are the location

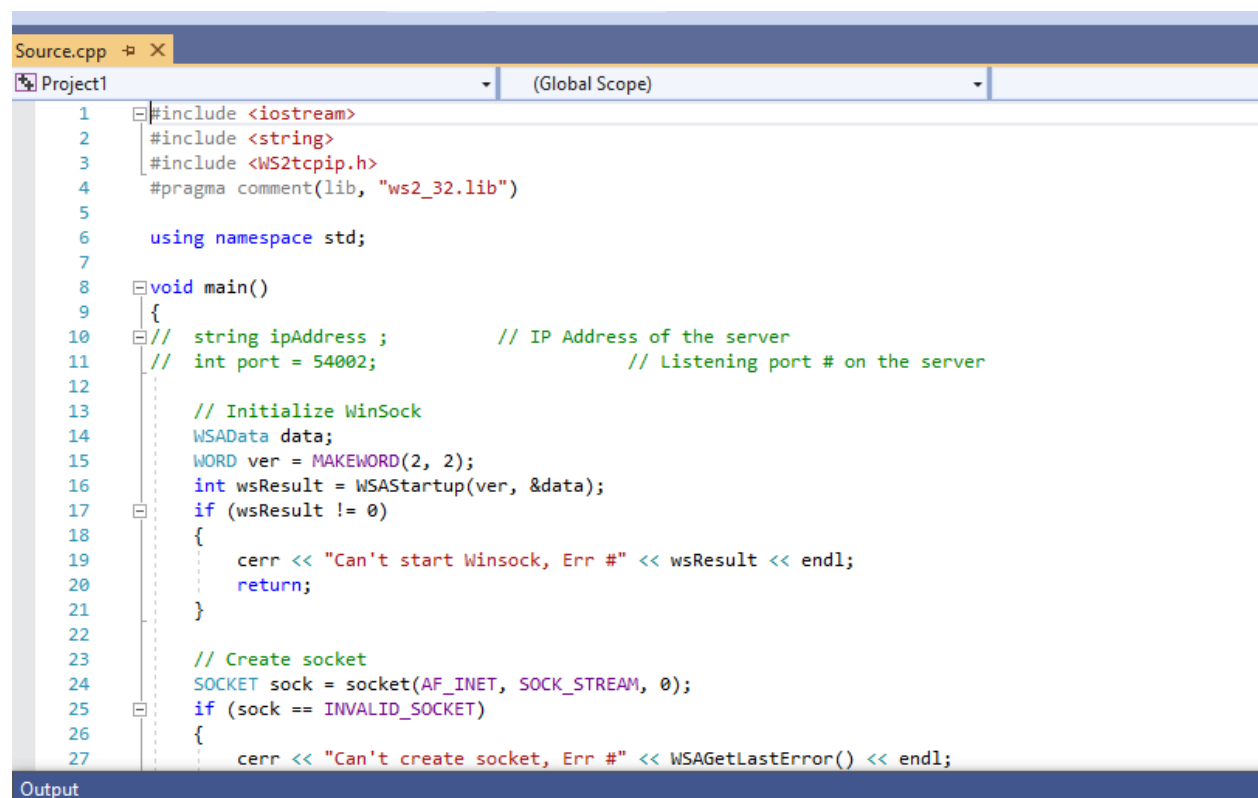
where information is sent and retrieved. Running a port scan on a network or server shows the ports (receiving information) are open and listening, as well as showing the existence of security devices such as firewalls between the sender and the target. This process is known as fingerprinting.

It is also useful for network security testing and the strength of the firewall of the system. It is also a common reconnaissance method for attackers finding a weak point of entry to break into a computer because of this feature.

The services provided by ports vary. They are numbered from 0 to 65535, although it is more frequent to use those ranges. Ports 0 to 1023 are classified as well-known ports" or regular ports, and Internet Assigned Numbers Authority services have been assigned to them (IANA). Some of the most relevant ports and their allocated facilities include:

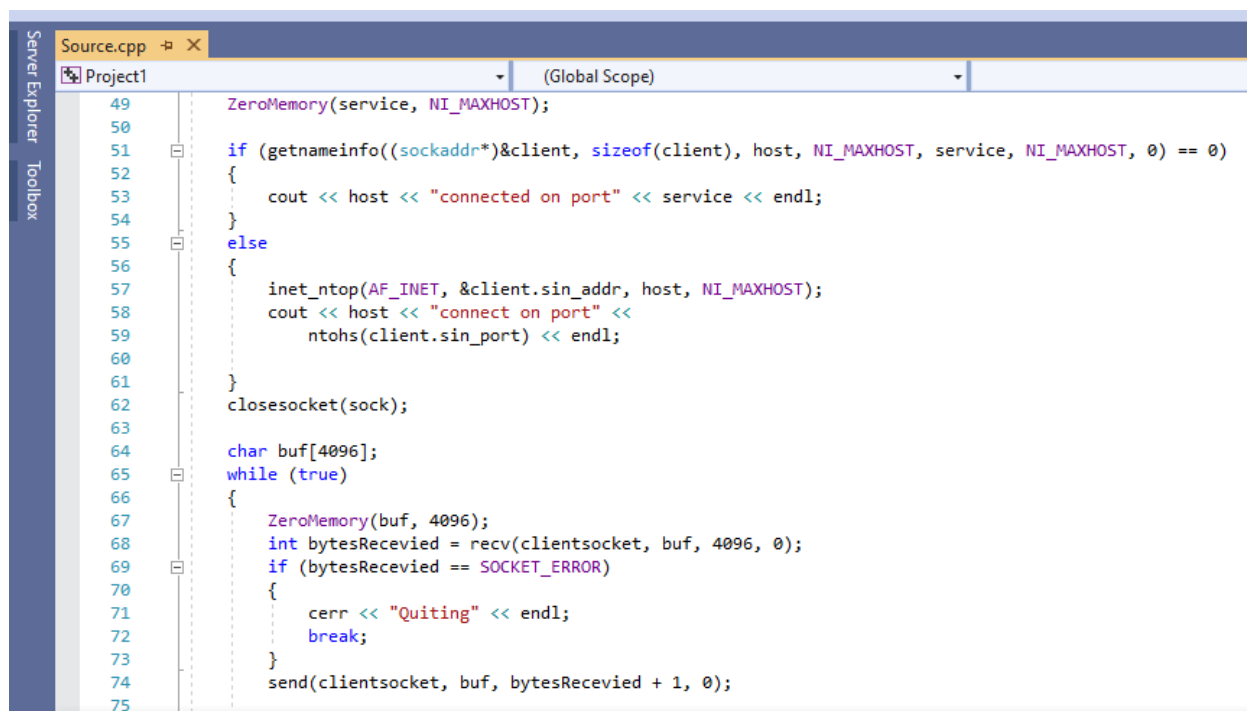
- Port 20 (udp) – File Transfer Protocol (FTP) for data transfer
- Port 22 (tcp) – Secure Shell (SSH) protocol for secure logins, ftp, and port forwarding
- Port 23 (tcp) – Telnet protocol for unencrypted text commutations
- Port 53 (udp) – Domain Name System (DNS) translates names of all computers on internet to IP addresses
- Port 80 (tcp) – World Wide Web HTTP

Coding:



```
Source.cpp [X]
Project1 (Global Scope)
1  #include <iostream>
2  #include <string>
3  #include <WS2tcpip.h>
4  #pragma comment(lib, "ws2_32.lib")
5
6  using namespace std;
7
8  void main()
9  {
10     // string ipAddress ;           // IP Address of the server
11     // int port = 54002;             // Listening port # on the server
12
13     // Initialize WinSock
14     WSADATA data;
15     WORD ver = MAKEWORD(2, 2);
16     int wsResult = WSStartup(ver, &data);
17     if (wsResult != 0)
18     {
19         cerr << "Can't start Winsock, Err #" << wsResult << endl;
20         return;
21     }
22
23     // Create socket
24     SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
25     if (sock == INVALID_SOCKET)
26     {
27         cerr << "Can't create socket, Err #" << WSAGetLastError() << endl;
```

```
27     cerr << "Can't create socket, Err #" << WSAGetLastError() << endl;
28     WSACleanup();
29     return;
30 }
31
32 // Fill in a hint structure
33 sockaddr_in hint;
34 hint.sin_family = AF_INET;
35 hint.sin_port = htons(54002);
36 hint.sin_addr.S_un.S_addr = INADDR_ANY;
37 bind(sock, (sockaddr*)&hint, sizeof(hint));
38 listen(sock, SOMAXCONN);
39
40 // Connect to server
41 sockaddr_in client;
42 int clientsize = sizeof(client);
43
44 SOCKET clientsocket = accept(sock, (sockaddr*)&client, &clientsize);
45 char host[NI_MAXHOST];
46 char service[NI_MAXHOST];
47
48 ZeroMemory(host, NI_MAXHOST);
49 ZeroMemory(service, NI_MAXHOST);
50
```

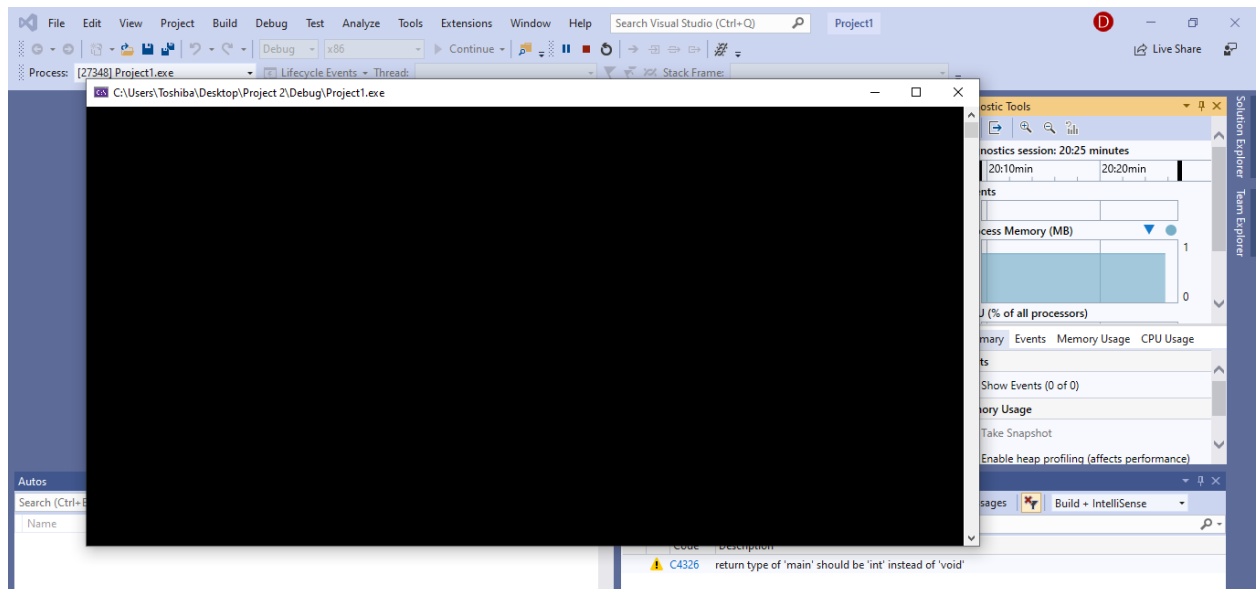


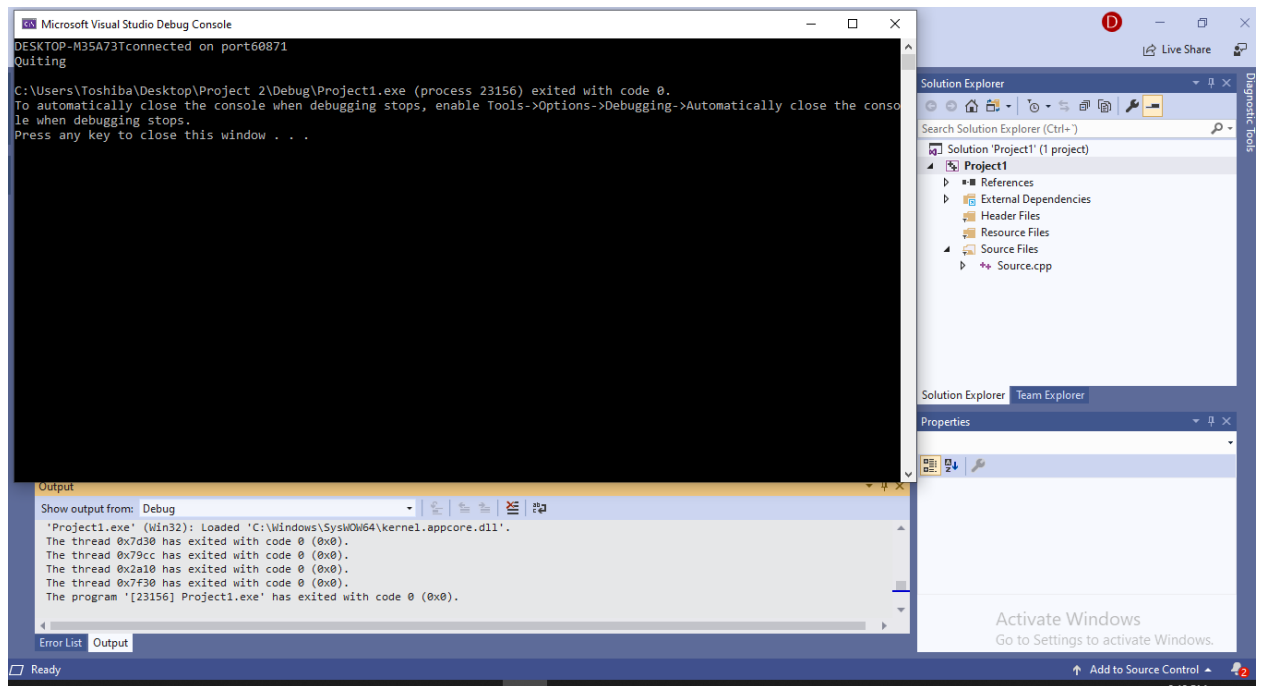
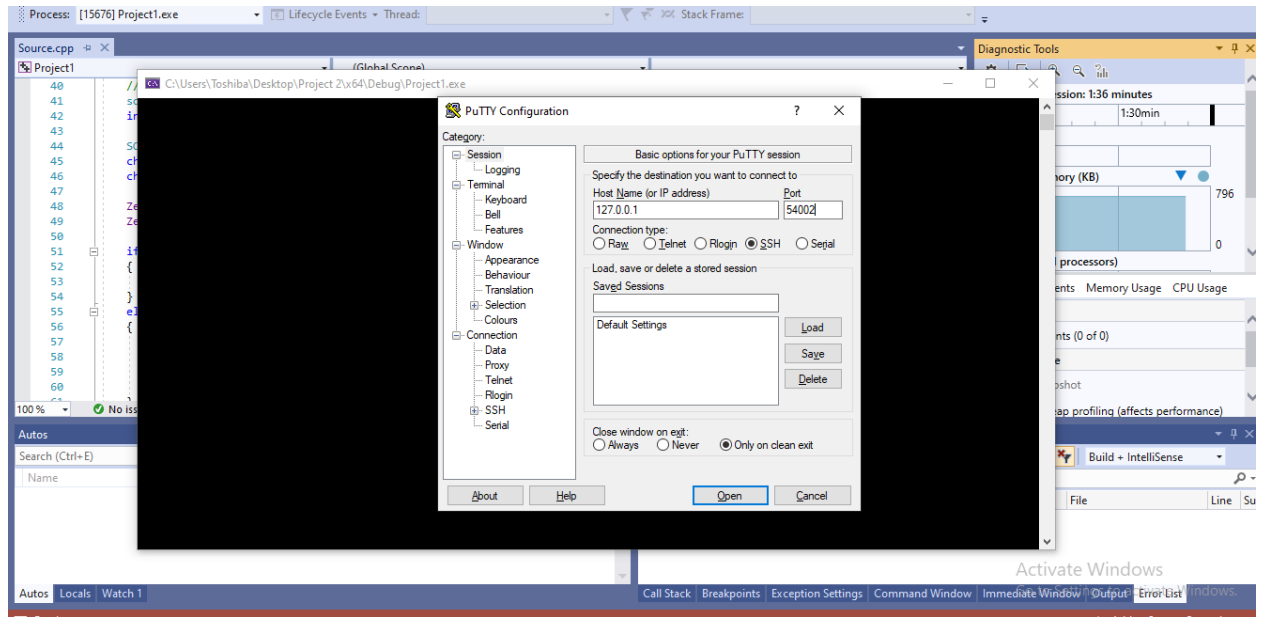
```
49     ZeroMemory(service, NI_MAXHOST);
50
51     if (getnameinfo((sockaddr*)&client, sizeof(client), host, NI_MAXHOST, service, NI_MAXHOST, 0) == 0)
52     {
53         cout << host << "connected on port" << service << endl;
54     }
55     else
56     {
57         inet_ntop(AF_INET, &client.sin_addr, host, NI_MAXHOST);
58         cout << host << "connect on port" <<
59             ntohs(client.sin_port) << endl;
60     }
61     closesocket(sock);
62
63     char buf[4096];
64     while (true)
65     {
66         ZeroMemory(buf, 4096);
67         int bytesReceived = recv(clientsocket, buf, 4096, 0);
68         if (bytesReceived == SOCKET_ERROR)
69         {
70             cerr << "Quiting" << endl;
71             break;
72         }
73         send(clientsocket, buf, bytesReceived + 1, 0);
74     }
75
```

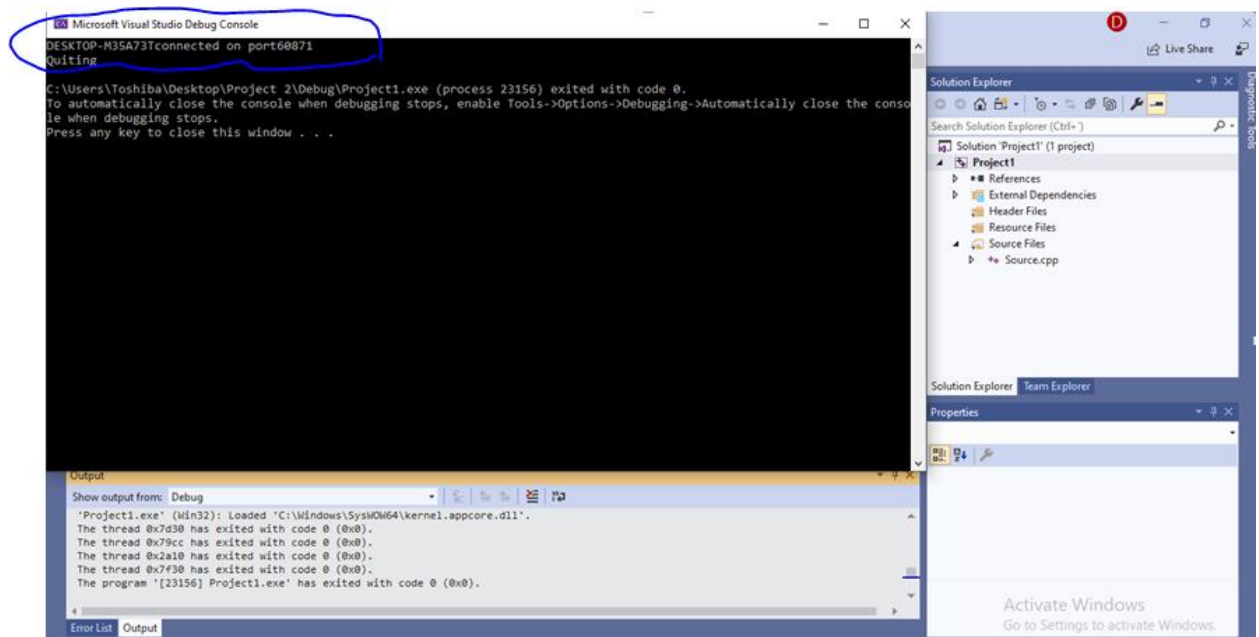


```
70     {
71         cerr << "Quiting" << endl;
72         break;
73     }
74     send(clientsocket, buf, bytesReceved + 1, 0);
75
76 }
77 closesocket(clientsocket);
78 WSACleanup();
79
80 }
81
```

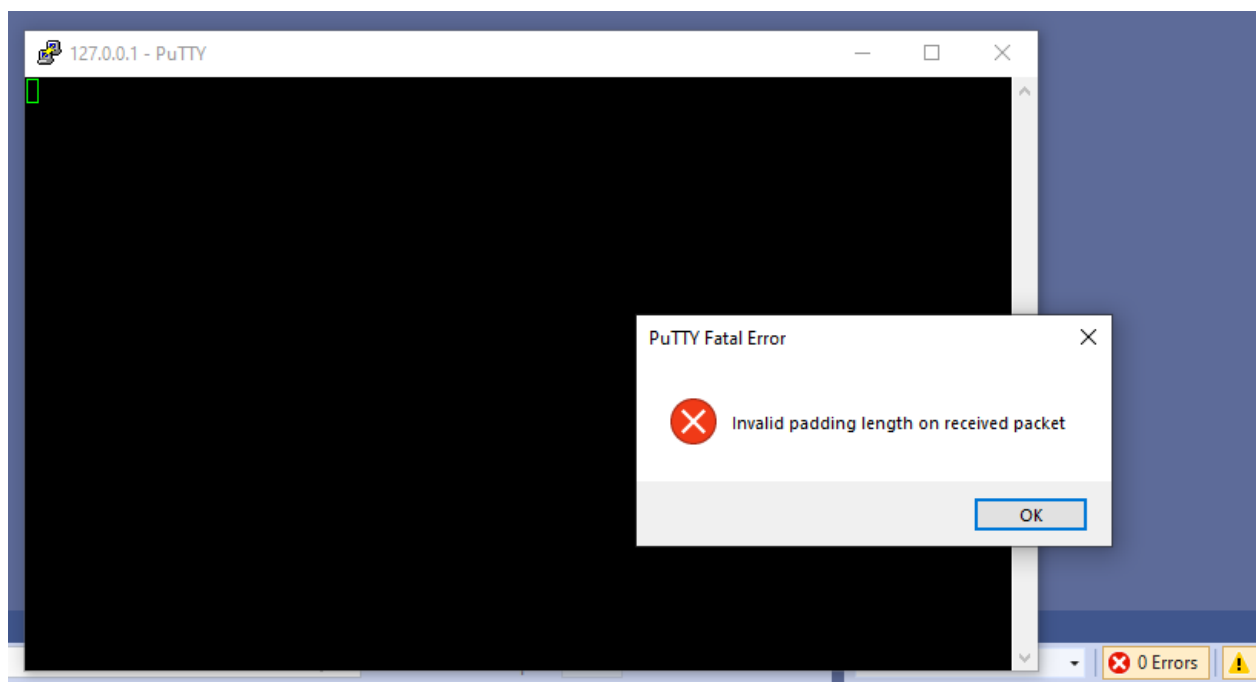
Results:

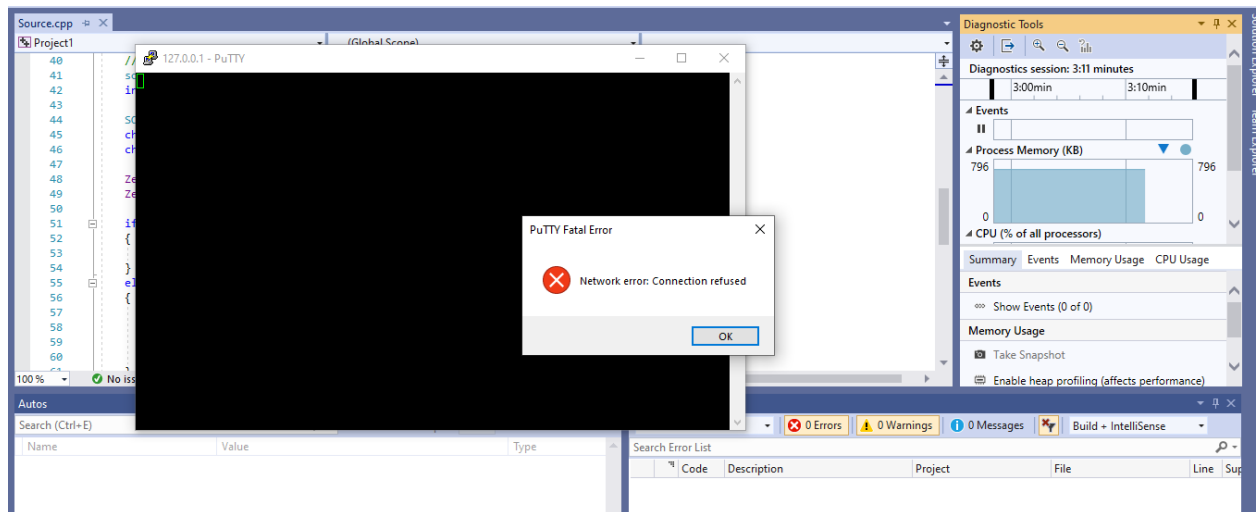






Problems:





This was occurred due to error in the port number.

Conclusion:

Therefore, from this experiment we had the clear understanding of TCP Port scan. We are able to understand the comprehension of the working concept of C/S contact mode and port scanning by finding the TCP service opened by the remote host understands the job opened by the remote host.