Designing a protocol for port scanners and analyzer

Name : Saif Ullah Khan
Roll No **:** 1802012
Submitted to: Dr. adnan Iqbal

Date : 10-June-2021

Abstract

Technology is growing day by day. In era of computer security, PORT scanners play a very important roles. These application detects weather a specific port is open or closed and also provides us various information about these port services.

In this report, I am designing a new protocol for port scanners application that will run on application layer and will use TCP on network layer for communications. This protocol is client-server based, the host will act as client and the target machine will act as server side. Client will send TCP connection request packets and note the server response. Server responded? Port is open and vice versa. After connection, client can send other messages that will request additional information about target system OS and that port services. These information will be send to client and analyzed for further knowledge.

Table of Contents

Introduction

In the era of modern technology, Cyber security field is very much important because all of our assets are stored online. By assets I mean our personal information like information on twitter, Facebook, Instagram, google accounts and so on. There could be sensitive information like credit card information like we used for online purchasing. All of these information need to be secured. Now if you are connected to an internet, believe me you are not safe! There is a lot of ways, someone get access to you computer and stole these information. Going deeper, for stealing these information, those hackers need some entry points in your system so that they can exploit your system. This entry point is called PORT. If there is open ports in your system, they can exploit your system by establishing a connection using these ports.

Every system have so many open ports that provides different services like port 8080 for HTTP and so on. Before entering into any system we need to know some information about the port and host system operating system. These information be like port is open/closed/filtered and which operating system is running on that system. This kind of information is very much important because without this, it will be very hard to get in into the host system. (You will have a bad story to tell your jail mates :))

In this report I am designing a new protocol for port scanning. Although it is very simple and basic but the main idea is to understand the working of PORT scanners so that we can make some better, fast, reliable protocols. In this design, the client machine will send TCP connection request to the host machine and note down the response. If the host replayed with SYN/ACK response, it mean there is a live machine with an open port. If the host refused the request, it mean the port is closed but there is a live host. If there is no response with that specific time limit, we will assume the request is filtered out. We are considering that as we request the host machine to provide its OS or PORT information, it will provide us (we are not considering the security issues like injection payloads for gathering information)

Functional Requirements:

- Client machine can send SYN packet request to target machine on specified port.
- The Target system must respond to the SYN message from the client. (in case of filtered packets, it doesn't respond)
- Target system respond with SYN/ACK packet.
- Client can specify a PORT range for scanning.
- Client request for PORT additional information and target system returns this information.
- Status of port and OS information is returned in TCP message from target machine.
- The application will send SYN request to specific port, received the response and analyze the response to check weather the port is open, closed or filtered.

Non-Functional Requirements:

- In case of filtered packets, no response will receive.
- Any message data will be send in text format
- User will be able to see time for each port checking process.
- There will SYN, ACK, RST flags so that machine can differentiate the packets of request, response and communication endpoint.

APPLICATION STRUCTURE

The application structure will client-server based. The host machine will initiate the process/communication and the target system will just respond to its request.

At one instance, There could be more than 1 clients who are pinging ports on target machine/server.

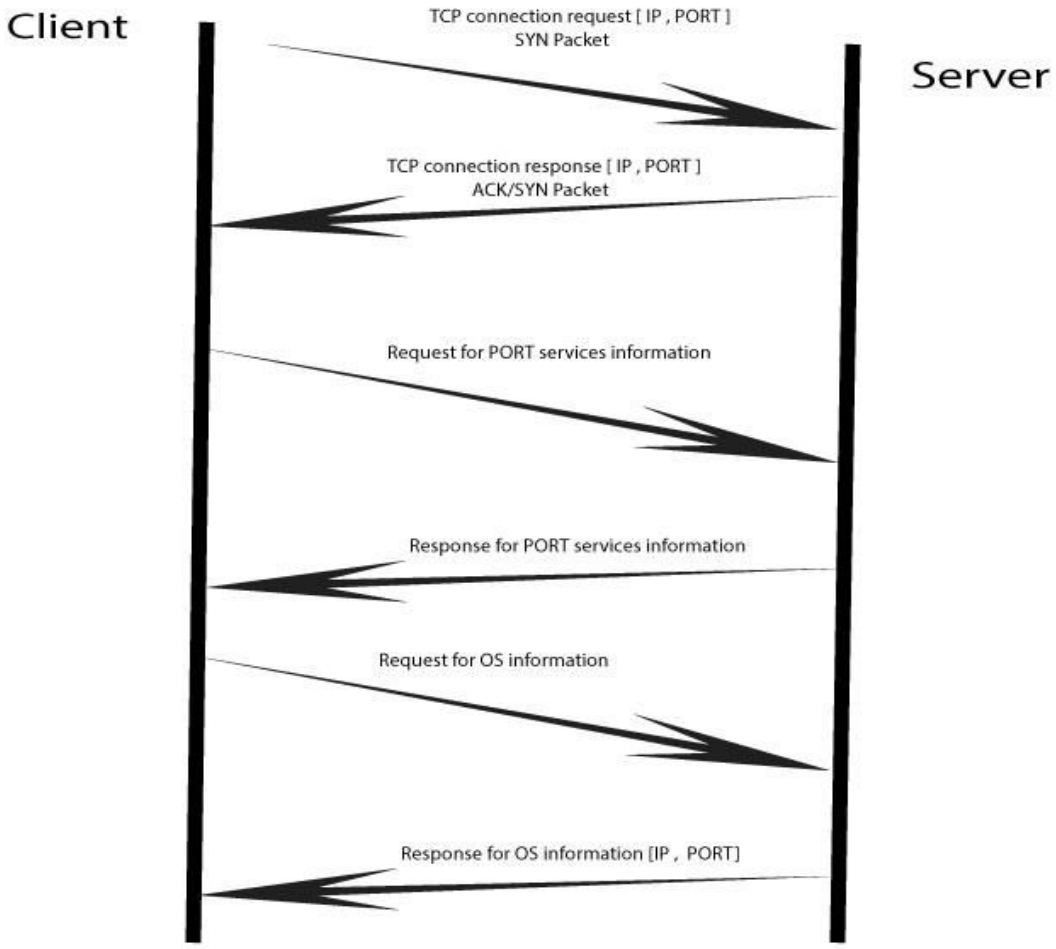


Fig. Request – response cycle.

THE PROTOCOL

Pattern of communication:

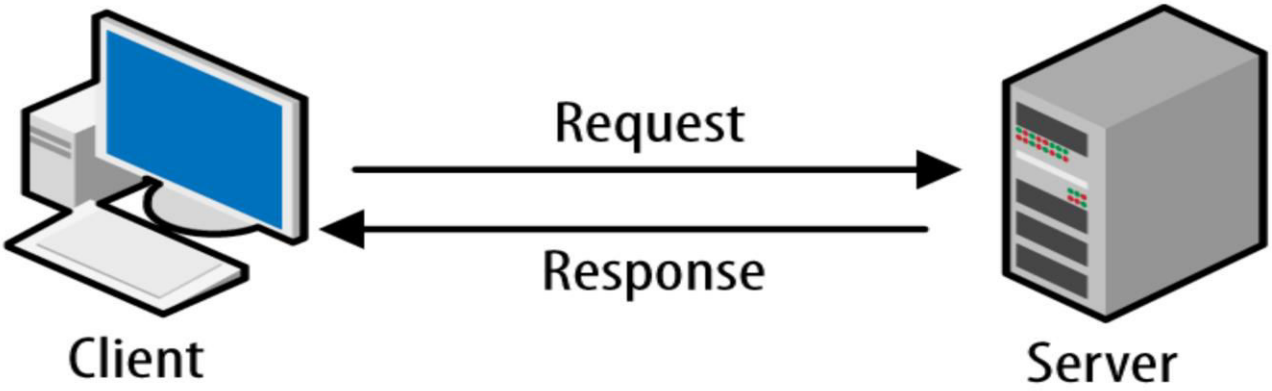
The pattern of communication will client-server based. The host machine will initiate the process/communication and the target system will just respond to its request.

At one instance, There could be more than 1 clients who are pinging ports on target machine/server.

After the communication is completed (we will get information about the PORT status).

Application will send another TCP request to get extra information about PORT, Like what kind of port is it? What kind of services it provides?

The target machine will then end this information using this TCP connection. Application will then send another packet and request information about OS. Target machine will full fill this request.



So to get complete information about target system we are making 3 TCP connection.

Patterns of transmission:

The transmission pattern will be many-to-one. The server can respond to many clients(Host machines) at same time. For example, host H-1 having address 192.168.0.0.1 is requesting TCP connection on port 4545 and host H-2 having address 192.168.0.0.2 is requesting TCP connection in port 35 at same time. The server will respond to both hosts.

Message format:

Message format will be text-oriented, because we don't need security for this application. As we are just sending request and wait for its response, we are good with the text message format. We are just making a connection between server and client so the text/body section need no special encryption like encoding the message into binary formats etc.
it will be simple as “Hello! I want to connect with port 67”

But for the other messages we will be using binary formats, which mean messages will be encoded into binary formats.

Message types

The message will be command type.

- We will be initiating the communication
- we will be requesting for the status of port

  Example: Hey! I want to connect with PORT 76. is it open or closed?

The message will be Data type.

- We will be requesting for information about PORT and OS.

  Request part will be command type message and information part will be Data type messages. (That will contain the information which are received from the target.

Message syntax

- Message type field will contain weather it is command message or data message.
- Body size will indicate the message length
- The command parameters
  - connection message flag or data requesting message (GET request).
  - The Source port & IP
  - The Destination port & IP
- Actual message

messages structure

SRC_ADDR: (IP, PORT)	DES_ADDR: (IP, PORT)
• 2 Flags: 1. Connection=0, GET=1 2. connection =1, GET=0 • Encoding format: “Binary’’	
Message Body Body will contain either dummy message or Requesting information messages or return information	

The connection flag will indicate weather the packet is connection related and the GET will indicates that client is requesting for information or server is sending information.

In the response if connection flag=1, It will determine it is SYN/ACK response
If the GET flag=1, It will determine it is the information request/Response between host and target system.

Message semantics

We have discussed two kinds of messages.

1. Simple connection TCP messages
2. Data requesting/receiving messages

   The first one is connection messages which will make connection between server and client. This is will also give some useful information like PORT is open or closed.

   The second one is data requesting/receiving messages.

   Both the messages contains some headers (SRC_ADDRESS, DES_ADDRESS), useful flags (variables) that will indicate the message is connection type or data request/response type. There is another field which will give information about the data encoding formats. The last section is the message body. The body will contain the message information. In case of connection, this will be empty. In case of requesting information this will contain the commands/requesting services names etc. In case of response This will contain all the data about PORT and OS that is sent by the server.

Additional notes for developers:

Now for developers using this application protocol, they can set the scan time like giving the -F or -Q scanning flags. The -F flags will indicates the user wants full network scan and will search for all ports in host system. This will show some messages like “this scan will take some extra time” and at the end the result is displayed.

The -Q flag will resides for quick scan, it will search for 1000 popular ports on host machine.

References :

- **https://nmap.org/book/osdetect.html**
- **https://www.techrepublic.com/article/exploring-the-anatomy-of-a-data-packet/**
- **https://dl.acm.org/doi/abs/10.1145/505733.505737**
- **https://searchnetworking.techtarget.com/definition/TCP**