Owen Burek

CMPT 220L-200

Professor Rivas

May 10, 2017

Project Milestone

Abstract

This project will demonstrate the creation of a port scanner that will check on the status of all network ports on a system, given the IP address. This program will use object oriented programming and the Socket Java library in order to complete this task.

Introduction

I want to create a port scanner in order to better understand the networking tools within Java, and programming techniques using networking libraries as a whole. In addition, I want to practice object oriented programming skills. I will begin by running through my driver program and my Comp class, and proceed to explain how they interact with the Socket class.

Detailed System Description

The system initializes a unique object of type Comp for each time it wants to scan a system. This object, which will contain both preliminary and conclusive data on the system being scanned such as IP address. It will have methods for getting, setting, and showing the runtime of the program in milliseconds. It will also have a function to scan a port of the system. The “driver” class (PortScan.java) will iterate through each and every port, creating a object of type Socket for each. Should the program successfully connect to a port, it will display to the user as open. Otherwise, it will

continue to the next port all the way to port #65535. After the scan method attempts to connect to a port, it will close the connection. Before and after the scanning loop, the program will start and stop, respectively, a timer that will calculate the number of seconds that has elapsed since the beginning of the program. At the end of the program, using the Comp class's showRuntime function, it will calculate and show how long it took to scan the system.

Literature Survey

From what I have seen, other port scanners either are limited in the scans they port, or use threading in order to speed up the scan time. Some also use functional programming to perform the scan rather than an object oriented programming.

User Manual

Users can input any private or public IPv4 address to scan for open ports in that system. Most IP addresses will block the scanner, causing it to do nothing. Should the scan prove successful, however, the user should still be cautious as to what they do with the information they glean from this program. Exploiting a system using this is not proper use, and thus constitutes a misuse of the software.

Conclusion

This project will, by using the Socket library and my Comp class, scan a system. The Comp class will store all relevant information regarding the scan, including the status of all ports on the system, as well as how long the scan took to complete. By proving empirical evidence, I could try to optimize this process more efficiently, so that it can be applied to a much larger scale.

UML Diagrams

(Socket from java.net.Socket)

Socket
close() connect() Socket() toString() getPort()
Comp
runtime:int Comp() setRuntime(rt:long) getRuntime():long showRuntime() scan(port:int, socket:Socket)
displayPorts()