Liam Powell

Port Scanner in C

A port scanning application written in C
Liam Powell

Application Description

Scanning for open ports on a device is an important step in maintaining device and network security. This program comes with several different methods of scanning to determine which ports are open across devices on an internal network. An internal network scan can help to identify methods of access by which an attacker could move from device to device once inside the network. By attempting connections on specified or generated network addresses, this program will report on the ports the connection was accepted on, depending on the scan type.

Program Usage

This program was designed for use on Apple M1 Mac devices with Xcode installed. As such, operationality is not confirmed for other devices and OSs. A full command list and examples will be listed below.

Scan Types

Default Scan

The default scan function attempts to create a connection to generated network addresses over port 80. This scan is designed to display the devices that have a common and easily attacked port open to the internal network. This scan is not designed to provide in-depth information or to report ports other than port 80.

Full Scan

The full scan function generates a report on the status of the 20 most commonly attacked ports on a specified network address. The report is generated in the form of a text file located in a folder called Scans. Each scan is saved with the date and time included in the filename to make searching easier. The report also provides insight on the ports that were found open, including information on what the port is used for and if it should be closed.

Select Scan

The select scan function is used to examine if a specific port is open on a specified network address. This scan is the quickest of the three, but provides no other feedback aside from the status of the port.

Interface, Environment, and Execution

This program is designed to be used entirely through the Terminal rather than the C console. There is no built-in interface included in the program. The idea is to use the program as any other Terminal or Console command.

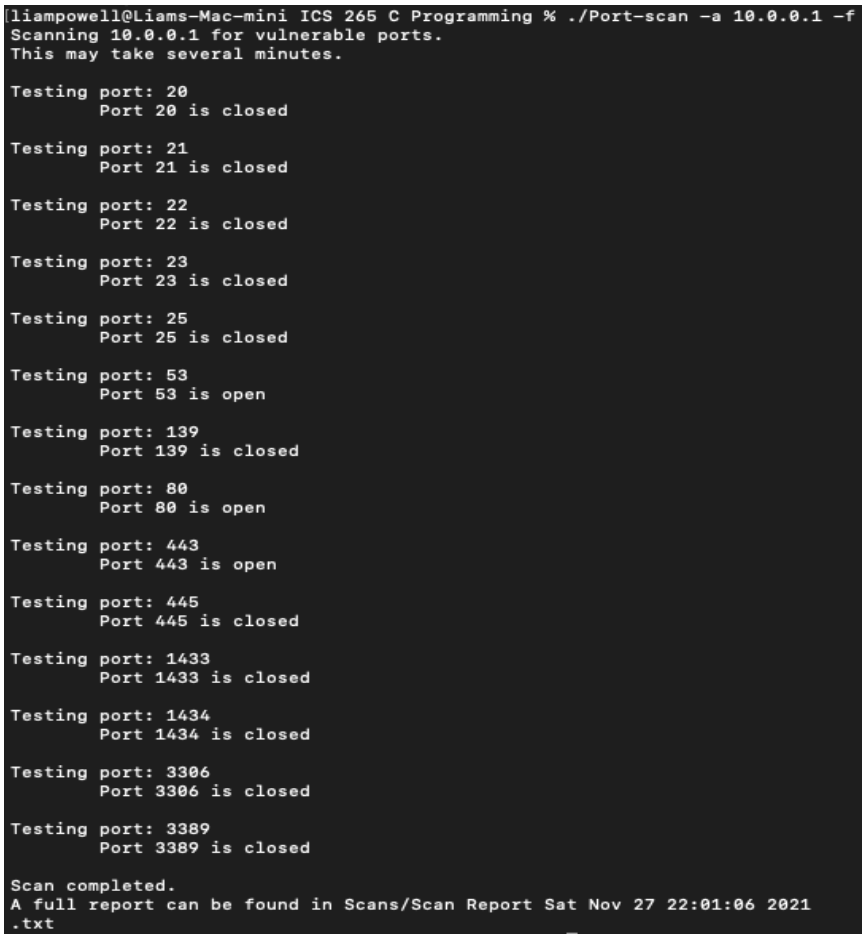
Usage is in the form of:

./Port-scan -[a,p] [string] -[h,d,s,f,]
- -h will open a help menu
- -d signifies a default scan
- -s signifies a select scan
- -f signifies a full scan
- -a designates an address to be used
- -p designates a port to be used

Example execution:

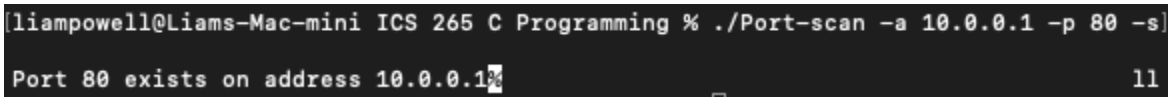
Full scan:

./Port-scan -a 10.0.0.1 -f



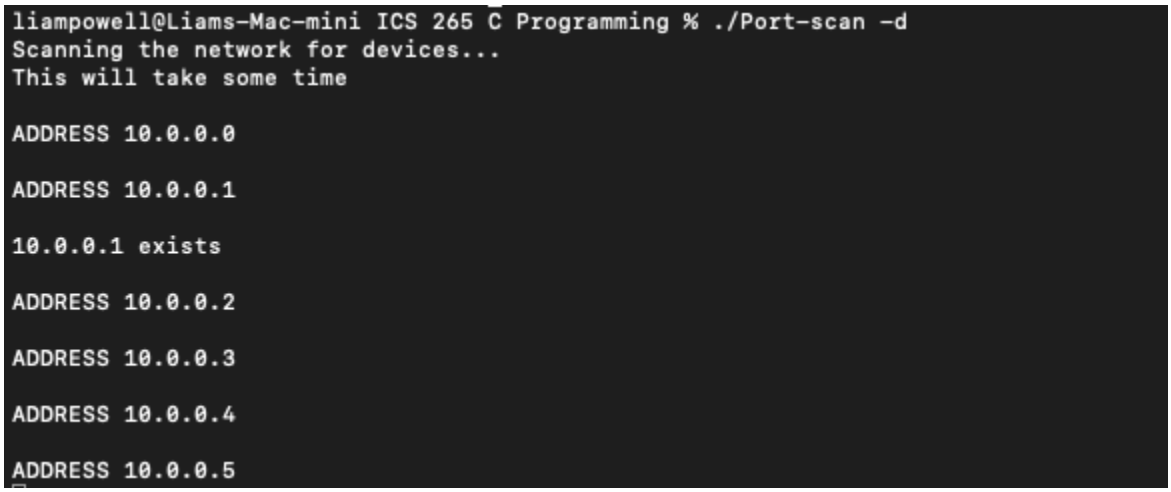
Select scan:

./Port-scan -a 10.0.0.1 -p 80 -s



Default:

./Port-scan -d



Program Structure

The source code is commented extensively, so this section will cover broad ideas rather than specifics.

Functions

socketCreate

socketCreate is the function that handles the creation of the socket. This function also includes error checking and will provide an error message if the function fails. On failure, the program aborts.

socketConnect

socketConnect takes the created socket and attempts to establish a connection to the address and port specified. It accepts the socket, address, and port as variables and returns a value that is used to verify functionality. If the returned value is not expected, the function has failed.

defaultScan

defaultScan is the function that handles the default scanning option. It iterates through every possible IPv4 address within the ranges of 10.0.0.1 and 10.255.255.255. If a connection can be established to the generated address, the program will print out an acknowledgment that the address exists.

fullScan

fullScan is the function that handles the report creation and scanning for the full scan option. The report is date stamped and sorted into a folder called Scans that is created within the directory the program is run from. The report also includes information on the ports found open and writes that information to the file it creates. fullScan only accepts the specified address.

selectScan

selectScan handles the select scanning option. It accepts both an address and port, and then attempts to establish a connection to the specified device.

help

The help function outputs information on the program.

main

The main function handles the command line options using getopts and passes the information to the necessary functions. If no option is provided the program aborts. If an unrecognized option is provided, the help function is displayed.

Improvements and Difficulties

The program was originally written midway through the class when I hadn't been exposed to certain elements such as structures and stacks. This program could have benefitted from these elements and would have seen improvements and usability upgrades if they had been included. Currently, the speed of the default scan is rather slow, as there is no functioning timeout system for connections. The program will wait to timeout on a connection before attempting the next address. A massive improvement would be in the form of a specified timeout that forces the connection to close after a user specified time has passed. Creating the socket array used in the program was also challenging. Ensuring that a socket was created and destroyed for each address took some figuring out.

Conclusion

The program works as originally intended, but could use additional functionality and usability. The idea of creating a port scanner originated with my networking class. I wanted to combine both classes together and create something that would use concepts from each class. I think that I reached that goal, and that overall, it was successful.