C++ GUI Network Scanner

CSC 388 Secure Computing

Fall 2019

Braden Bagby

Dr. Katangur

December 5, 2018

Missouri State University

Network scanning, like many concepts in computer security, can be used by both hackers and network administrators. It is done to find what machines are connected to a given network, information about those machines, and what services those machines might be running. Port Scanning is a type of network scanning that attempts to discover what ports are open on a device connected to a given network. There are many different port scanning techniques that have their own advantages and disadvantages. For this project, I created a GUI network port scanner written in C++ that implements SYN, TCP Handshake, FIN, and XMas scans, as well as allowing the user to perform their own custom scan. I will be discussing these port scanning techniques, their advantages and disadvantages, and how I implemented them in C++ with a GUI interface.

GUI in C++ today is surprisingly easy. Qt is an open source toolkit that allows you to create 'widgets' in C++ using XML. Yet, with Qt Creator (the IDE thats built specifically to work with Qt), one can design widgets by simply dragging and dropping, which will automatically generate the XML for you. Signals and Slots are Qt's way of allowing event driven programming, making it easy to have your widgets interact with your code, and your code to interact with your widgets. A slot is simply a function called when a signal is released from anywhere in the program specified. I used Qt and Qt Creator to design the interface for my network scanner, giving the user input fields for the IP address, ports, log level, and scan type. This worked well with integrating the UI with the scanning code.

The scanning code was all created under one class, 'Scanner'. Scanner is a singleton as only one instance of it should initialized at a time. Its most important function is the Scan

function, which takes a scan type and a list of IP Addresses and ports to scan. This function loops through each IP and each port and calls the correct internal scan function depending on the scan type argument. Each internal scan function is called in a separate thread in order for the computer to run scans in parallel and increase the speed of the scan. Each internal scan will release a Qt Signal of its result, which will update the UI. When the whole scan completes and all the internal scanning threads have completed, Scanner releases a Qt Signal to notify the UI of a scan complete as well.

The easiest type of scan to implement is TCP Handshake. This scan does exactly what you may expect. It attempts to complete a TCP connection, a success being the port is open and a fail interpreted as a closed port. Because you are simply doing a TCP Handshake, to implement this in C++ all you must do is use system internet sockets to attempt to connect using SOCK_STREAM.. This means the user will not need to have root capabilities, as SOCK_STREAM sockets are available to all users. The downside to this type of port scan is the speed. It must attempt to complete the entire TCP Handshake for every port it is to scan.

All of the other port scanning techniques used require a custom TCP packet and raw sockets. I used a library called ‘Tins’ to help in the construction of these packets. With Tins, you create PDUs (Protocol Data Units) and encapsulate PDUs within PDUs. This means you can wrap a TCP PDU within an IP PDU within an Ethernet PDU and send as a raw packet. This helped in constructing my custom TCP packets, setting the correct flags, and sending through the network.

The most popular port scanning technique is the SYN scan. It sends the first packet in a tcp handshake by setting the SYN flag, and uses the response to detect if the port is

open/closed/filtered. Because you are only sending part of a TCP handshake, this type of scanning requires root privileges to create custom packets. If the response has a SYN and ACK flag the port is open and a service is running. If the response has either SYN or ACK flag but not both, the port is definitely open. If the response has the RST flag on, the port is closed. Finally, no response is considered as the port being closed/filtered after multiple retransmissions. This program does 4 retransmissions before considering a no response as closed. Since you are not completing an entire TCP handshake, this port scanning technique is much faster than TCP handshake scanning.

FIN and XMAS scans are very similar, so much in fact that they result in the same behavior. A FIN scan simply sets the FIN flag of a tcp packet, while a XMAS scan sets the FIN, PSH, and URG flags of a tcp packet and is called XMAS scan because it lights the packet up like a christmas tree. These scans work due to the rules set in the TCP Protocol stating that If a device is up and port is closed, an incoming segment not containing RST flag results in device sending RST flag. These scans can be more stealthy than a SYN scan as some firewalls block TCP packets with the SYN flag set. Yet, this type of scan is unreliable. A response packet with an RST flag set is interpreted as the port being closed, but a no response is interpreted as the port being open. This can result in many ports being reported as open when in fact they are just unreachable.

The Network GUI Program also allows a user to run a custom scan and create a custom TCP packet. The user can select what flags to enable in the TCP packet, the number of packets to send, the destination, and port. He/She can then use the resulting response printed to the screen

to determine information about the scanned host. This allows a user to learn about port scanning in a lower level way while not having to worry about programming.

In conclusion, port scanning allows someone to discover what ports are open on the network. There are many different techniques a port scanner can use considering the importance of speed, stealth, and privileges on the scanning computer. My C++ Network GUI scanner allows a user to conduct a TCP Handshake, SYN, Xmas, and Fin scan. It also allows a user to send custom TCP Packets for his/her own inspection. A lot about what is on a network can be learned through port scanning, and it is important for a network administrator to know how to port scan as well as how port scanning works in order to keep the network secure.