**Port Scanners**

**Augustin U. Tran**

**CSCE 4550**

# What are Port Scanners?

Port Scanners are powerful tools that allow users to check which ports are open in a network. Hackers can use them to check for vulnerable ports while administrators can use them to both verify security policies and check for vulnerable ports.

Ports are usually generalized into one of three categories:

1. Open: The host responded back, indicating a service is listening on that port.
2. Closed: The host responds back denying connections to the port.
3. Filtered/Dropped: The host did not respond.

Open ports present vulnerabilities for the security and stability of both the service running on the port and the operating system running on the host. Filtered ports tend to not present vulnerabilities.

There are many types of scans including:

1. TCP Scanning
   1. The Port Scanner uses the UNIX connect() system call, completes the TCP three-way handshake, and immediately closes connection. If it fails at any point, an error is returned and the port is flagged as closed. It is useful in the fact that the user does not require any special privileges but using the OS network functions prevents low-level control. Furthermore, it is noisy method that allows services to log the sender’s IP address, easily raising alarms. Attackers generally avoid using this method of scanning.
2. SYN Scanning
   1. The Port Scanner generates raw IP packets that monitors for responses. Unlike TCP, SYN scanning does not complete a full connection and closes the connection before the handshake is complete. Not only is this far less noisy, it also allows full control of packets sent and timeouts for said packets.
3. UDP Scanning
   1. A connectionless protocol that responds back when a port is not open. There are problems in that firewalls can block those messages, making ports appear open. Generally not used by attackers because it is too easy to protect against.
4. ACK Scanning
   1. Unlike the above scanning techniques, ACK scanning does not check if a port is open or closed. Instead, it checks if a port is filtered or unfiltered, making it useful for checking for firewalls. It can be used along side the above techniques to obtain more information, finding not just open ports but unfiltered ports. This would be the equivalent of checking for alarms in a window or door.
5. Window Scanning
   1. Untrustworthy scanning technique. Generates packets and checks if the window field of the packet is modified. If the packet found an open port, it would modify its window field with 1’s, flagging the above condition. However, systems no longer support this implementation, causing packets to return with 0’s, making open ports appear closed.
6. FIN Scanning
   1. Since SYN packets are the focus of firewalls, FIN packets can sneak past without modification. Closed ports will respond to the packet while open ports ignore the packet.
7. X-MAS Scanning
   1. An unusual scanning technique that sends an equally unusual packet to ports. The reaction of the port will change an option bit in the header of the packet, revealing information about the host’s operating system.

As shown, there’s an immense variety of possible scanning techniques. The ideal type used varies on purpose, giving specific types, such as X-MAS scanning, red flags when seen by the defending network. I will be using the TCP scanning technique for my Port Scanner.

For attackers, Port Scanners are generally used to find vulnerable ports for attack. For them, it would be the equivalent of checking all the doors and windows of a house or company. If any ports are open and unmonitored, it is easy to get in and cause immense damage, stealing important information or sabotaging the business. When done the right way, port scans can even cause buffer overflow and DoS attacks.

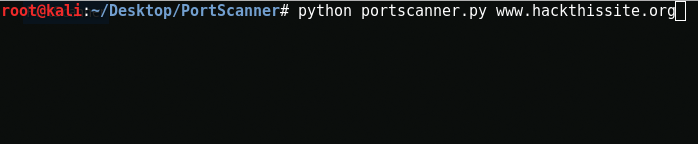
As such, Port Scanners are incredibly important tools to utilize and understand. By understanding how Port Scanners work, businesses can defend against attacks, preventing or bluffing information retrieved by scans. For my project, I built a simple Port Scanner that retrieves services using open ports. While NMap is a powerful tool with a Port Scanner already built in, I built this for the sake of education purposes. By building one myself, I can better understand port scanning, how it is used, and why we need it.

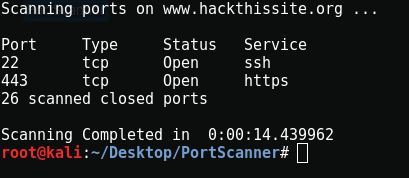
# My Project

As stated, I wrote my own Port Scanner for better understanding of how it all works to better apply tools like NMap. For simplicity, I used Python and ran the script on Kali Linux. I picked Python since I wanted to use sockets and I felt the most comfortable writing with sockets in Python. I used Kali Linux since I already had NMap installed on my Kali Linux system and I wanted to double check everything with NMap on the same machine.

My Port Scanner takes in a host and scans said host using sockets. The user may specify a range of hosts to scan. If not, the Port Scanner will scan just the most common ports. The scanner will then report back which ports are open, the service running in them, the protocol type, and the time it took to complete the above.

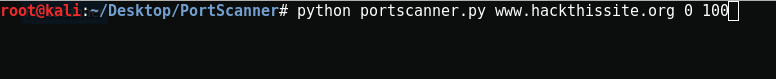
Here is an example scan of the common ports for www.hackthissite.org:

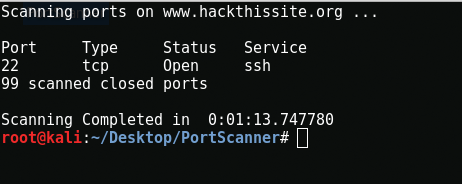




As of the time of this scan, only port 80 was open out of the ports scanned. As shown, port 22 was running ssh and is of type TCP and port 443 was running https and is of type TCP. This was verified with NMap as well.

Here is an example scan of ports 0-100 for www.hackthissite.org:





As of the time of this scan, only port 22 was open out of the ports scanned. As shown, port 22 was running ssh and is of type TCP. Also verified with NMap.

# Discussion

As shown, this Port Scanner is fairly fast and gets some quick and useful information. Knowing what ports are open and what is running on it already gives enough information to find what ports are open and unguarded, which is incredibly useful and important information for both attackers and businesses.

In application, say a business opens port 888 for testing their program. However, they forgot to close the port after testing. Attackers that find the open port can craft a TCP packet with a malicious string, encapsulate it in an HTTP packet and target the open port, exploiting potential vulnerabilities in the program running. As for the business, they could have realized the port was left open through regular scans. As you can see, scans this simple already can make or break entire attack operations.

Now, I did not meet everything I wanted in the proposal. I wanted to gather more information from found ports but ran into immense difficulty. In the end, the Port Scanner ended up performing what NMap’s default scan does but that is still a powerful tool. To improve, I could have added ACK scanning as well to check if the port is filtered or unfiltered, increasing the scanner’s capabilities.

# Conclusion

In the end, I learned immense amounts of information regarding scanning, how scans are done, and how they can be used. The sheer amount of types of scans and the ways to use each is astounding. Thanks to this, I now better understand TCP, UDP, and other protocols. I even learned the existence of X-Mas, idle, and null scans, which I probably would not have learned had I not done more research on types of scans for this project. Furthermore, thanks to regular testing, I became more familiar with common ports and which ports should be open in general. Even more, I am pretty sure I only scratched the surface of Port Scanning and its applications in the world.

I also learned how businesses defend against such attacks, how they can give false information and throw off scans to defend their network. Furthermore, I learned just how useful it is to attempt to make at tool that already exists. By making your own version, you can learn how to better utilize the tool in real world scenarios.