Port scanning with nmap

After discovering live hosts on a network using host discovery techniques, the next step is to determine which ports are open on those hosts and what services are running on those ports. This information can help identify potential vulnerabilities and attack vectors.

We will start with Port scanning first.

Port scanning is a way to check which ports on a computer are open and accepting traffic. It's like walking up to a house and trying different doors to see which ones are unlocked.

Like in host disovery, we also have various techniques to perform Port scanning in nmap.

- TCP Scanning

1. Full TCP Scan or TCP Connect Scan
2. Half TCP Scan or TCP Stealth Scan
3. Inverse TCP Flag Scan - Xmas Scan, FIN Scan, NULL Scan, Maimon Scan
4. ACK Flag Probe Scan - TTL Based Scan, Window Scan
5. IDLE/IP ID Header Scan

- UDP Scanning
- SCTP Scanning

1. SCTP INIT Scanning
2. SCTP COOKIE/ECHO Scanning

- SSDP Scanning
- IPV6 Scanning

TCP Connect Scan

Lets start with the TCP Scanning technique. The first one we have is the Full TCP Scan or TCP Connect scan.

A TCP Connect Scan is a basic port scanning technique that establishes a full TCP 3 way connection to determine if a port is open on a target system.

- Nmap sends a TCP SYN packet to the target port, just like it would for a normal TCP connection.
- If the port is open, the target responds with a TCP SYN-ACK packet, indicating it's ready to establish a connection.
- Nmap then sends a TCP ACK packet to complete the three-way handshake and open the connection.
- Once the connection is established, Nmap immediately sends a TCP RST (reset) packet to close it down.
- If the port is closed, the target responds directly with a TCP RST packet instead of SYN-ACK, refusing the connection attempt

To perform this scan, we will use the -sT flag.

```
nmap -sT IP
```

The main advantages of TCP Connect Scan are:

- It's simple to implement and doesn't require special privileges
- It works on all platforms, even those that block raw packet scans

However, the disadvantages are:

- It's slower and less stealthy than other scans, as it establishes full connections.
- It's more likely to be detected and logged by the target system

---

TCP Stealth Scan

Next, we will see Half TCP Scan or Stealth Scan.

A TCP Half Connect Scan, also known as a Stealth Scan, is a technique used to quickly determine which ports on a target system are open, without fully establishing a connection.

Here's how it works:

- **The Scan**: The scanner sends a TCP SYN packet to each target port, which is the first step in establishing a TCP connection.
- **The Response**: If the port is open, the target responds with a TCP SYN-ACK packet, indicating it's ready to establish a connection. If the port is closed, the target responds with a TCP RST (reset) packet, indicating the connection is refused.

To perform this scan, we will use -sS flag.

```
sudo nmap -sS IP
```

The thing to note here is that this scan does require sudo permissions to operate and if you are using an nmap scan with sudo without specifying -sS flag, it by default uses the stealth scan. If we don't specify sudo then it by default falls back to full connect scan.

Advantages of this scan:

- **No Full Connection**: Unlike a TCP Connect Scan, the scanner does not complete the full TCP handshake by sending an ACK packet. Instead, it just waits for the SYN-ACK or RST response.
- **Stealthy**: This technique is called "stealthy" because it doesn't establish a full connection, making it harder to detect by firewalls and intrusion detection systems.
- **Faster and Less Traffic**: TCP Half Connect Scans are faster and generate less network traffic compared to TCP Connect Scans, which makes them more efficient.

Inverse TCP Flag Scans

Inverse TCP Flag Scans are a type of port scanning technique that exploits a loophole in the TCP protocol specification to determine if a port is open or closed on a target system.

The Scan: Instead of using the standard TCP SYN flag to probe ports, nmap sends packets with unusual TCP flag combinations

- FIN Scan: Sets only the FIN flag
- NULL Scan: Sets no flags at all
- XMAS Scan: Sets the FIN, PSH, and URG flags.

The Response: According to the TCP RFC, if a port is closed and receives a packet without the SYN, RST, or ACK flags set, it should respond with a RST (reset) packet. If the port is open, it should silently drop the unusual packet.

Identifying Open Ports: By analyzing the responses from the target system, the scanner can determine which ports are open or closed.

- No response: Port is likely open
- RST packet received: Port is closed
- ICMP error: Port is filtered by a firewall or other device

XMAS Scan

Let's see all the scans one by one. Starting off with XMAS scan.

To perform the XMAS scan, we use -sX flag. It uses FIN, URG and PSH TCP flags in its packets.

```
sudo nmap -sX IP
```

The thing to notice here is that all the ports here are showing open or filtered. This is because, nmap is not able to confirm if the port are actually open or not. It is kinda confused here but we can conclude with this result that the ports are open.

Another thing with the XMAS scan is that it does not work on Windows machine. So, if you try to scan a windows machine using XMAS scan, it wont work due to different TCP/IP Stack implementation. It only works on particular Unix systems.

FIN Scan

Next, we have FIN Scan.

To perform the FIN scan, we use -sF flag. It uses only FIN TCP flag in its packets.

```
sudo nmap -sF IP
```

We got the same result for FIN scan also. One interesting thing about FIN scan is that it only work OSes that uses RFC-793 based TCP/IP Implementation. RFC are basically a document that consist information on how networks are developed and implemented. In real world pentesting, we dont have much to do with that, until we are researching on something related.

NULL Scan

Now we have NULL scan.

To perform the NULL scan, we use -sN flag. It only send an empty or NULL packet

```
sudo nmap -sN IP
```

Got the same results again for the open ports.

Maimon Scan

At last, we have Maimon Scan.

To perform the Maimon scan, we use -sM flag. It uses FIN/ACK TCP flags in its packets.

```
sudo nmap -sM IP
```

The key advantages of Inverse TCP Flag Scans are:

- They can bypass certain non-stateful firewalls and packet filters that only block SYN packets.
- They are slightly more stealthy than a standard SYN scan, as they don't complete full TCP connections.

However, these scans have some limitations also:

- They don't work reliably against all operating systems, as some systems don't follow the TCP RFC precisely.
- They can't always distinguish between open ports and certain filtered ports.
- They are less accurate than other scanning methods like SYN scans

---

ACK Flag Probe Scan

An ACK Flag Probe Scan is a technique used to determine if a target system has a firewall or filtering system in place.

Here's how it works:

1. First we sends a TCP packet to the target system with only the ACK flag set, along with a random sequence number. This is called an "ACK probe packet".
2. If the target port is open, it will respond with a TCP RST (reset) packet, indicating the connection is refused
3. If the target port is filtered by a firewall or other security device, the ACK probe packet will be dropped and no response will be received
4. By analyzing the responses (or lack thereof) from the ACK probe packets sent to various ports, the we can determine if the target system has a stateful firewall in place

To perform an ACK Flag Probe Scan using Nmap, you would use the `-sA` option. For example:

```
nmap -sA IP
```

TTL based

There are also two kinds of ACK Flag probe Techniques that are used to find open ports. These are - TTL based and Window Based ACK Flag Probe Scanning

First lets see what is TTL based Scanning.

But before jumping into that. lets understand what TTL means.

TTL stands for "Time to Live". It's like a countdown timer that tells how long something can live or exist before it has to go away.

Imagine you have a balloon that you blow up. The balloon has a certain amount of air in it, and that air can only last for so long before it escapes and the balloon deflates. The amount of time the balloon can stay inflated is like the TTL.

In computers, TTL works a similar way. When you send information over the internet, it gets broken up into little pieces called packets. Each packet has a TTL value that starts at a certain number, like 64 or 128.

As the packet travels from one computer to another, the TTL number goes down by 1 each time. If the TTL reaches 0 before the packet reaches its destination, the packet gets thrown away because it's too old.TTL helps keep the internet clean and organized. If packets could live forever, they might get stuck going in circles and clog up the whole system. The TTL makes sure old packets don't hang around too long.

Generally, the TTL values for Linux machines are 64 and for Windows machines this is 128.

Now coming back to the topic.

So, basically in a TTL based scanning, we first send an ACK probe and then analyze its TTL field value for the RST packets. If the TTL value of the RST packet on a particular port is less than the boundary value of 64, then that port is open.

To perform this scan, we use

```
sudo nmap -p 22 IP
```

-> Use wireshark to check for TTL value

Window based

Next, we have the Window based ACK Flag probe scanning method.

But first lets understand what is the windows size.

Imagine you have a bunch of friends over and you're all playing a game together. You have a bunch of cards that you need to share with your friends, but you can only give them a few cards at a time.

The number of cards you can give them at once is like the packet window size. It's the maximum amount of information that can be sent from one computer to another before the receiving computer has to say "I got it, send me some more!"

For example, let's say the packet window size is 5 cards. That means you can give your friends 5 cards, then you have to wait for them to say "Okay, I'm ready for more!" before you can give them 5 more cards.

If the packet window size is bigger, like 10 cards, you can give your friends more cards at once before having to wait for them to be ready for more. But if the window size is too big and you give them too many cards at once, they might get confused and drop some of the cards!

So the packet window size is like the number of cards you can give your friends at a time when you're sharing information. It helps make sure the information gets shared smoothly without anyone getting overwhelmed. The computers use this system to make sure they can send and receive information as fast as possible without any problems.

Coming back to the topic now.

In this scanning technique, we first send an ACK probe and then analyze its Window size value for the RST packets. If the Window size value of the RST packet on a particular port has a non zero value, then that port is open.

To perform this scan, we will use the -sW flag.

```
sudo nmap -sW IP
```

Now the thing to note here is that some of these scans are not working on our target system. This can be because of the target OS, TCP/IP Implementation or some other factors. These types of scans are not generally used in real engagements as they depend on some very specific stuff.

The key advantages of an ACK Flag Probe Scan are:

- It can bypass certain types of firewalls that only filter SYN packets
- It is more stealthy than a full TCP connection attempt

- It can map out the firewall ruleset to identify which ports are filtered

However, it has some limitations:

- It cannot determine if a port is open or closed, only if it is filtered
- It relies on the target system responding to the ACK probe as expected
- It is less accurate than other scanning techniques like SYN scans

---

IDLE/IPID Header Scan

An IDLE/IP ID Header Scan is a clever way to scan a target system's open ports without being detected, by using a third "zombie" system as a proxy.

Here's how it works:

- The attacker first finds a suitable "zombie" system that has a predictable IP ID number. This means the IP ID increments by a known amount each time a packet is sent.
- The attacker sends a SYN/ACK packet to the zombie to determine its current IP ID value. The zombie responds with a RST packet containing the IP ID.
- The attacker then spoofs a SYN packet to the target system, using the zombie's IP address as the source.
- If the target port is open, it will respond to the zombie with a SYN/ACK packet.
- The zombie, thinking the SYN/ACK is a response to its own earlier SYN/ACK, will send a RST packet back to the target. This causes the zombie's IP ID to increment.
- The attacker then checks the zombie's IP ID again. If it has incremented by 2, then the target port is open but if its value has only incremented by 1. Then the target port is eventually closed.

So in essence, the attacker uses the zombie system's IP ID as a "sensor" to detect if the target responded to the spoofed packets, without the target ever seeing the attacker's real IP address. This makes the IDLE/IP ID scan extremely stealthy, as the target only sees packets coming from the zombie system. It's a clever way to bypass firewalls and avoid detection when port scanning.

However, this technique is a little difficult to implement in real life nowadays. As the main requirement of the technique is the zombie system, which is very hard to find nowadays, in my opinion. But nevertheless it is a great technique and you should have it in your arsenal because you never know when it might comes in handy.

UDP Scanning

Moving on, we have reached to the UDP Port Scanning section now.

UDP port scanning is a way to check if a computer is listening for and accepting UDP traffic on specific ports.

Here's how it works:

- The scanner sends a UDP packet to each target port on the computer. This packet is like knocking on a door to see if anyone is home.
- If the port is open and a service is listening on it, the service may send a response back. This is like someone answering the door.
- If the port is closed, the computer may send back an ICMP "port unreachable" message. This is like someone saying "wrong door, no one here".
- If no response is received at all, it could mean the port is open but the service didn't respond, or it could be filtered by a firewall. It's like knocking on a door and hearing no answer, so you don't know if anyone is home or if the door is blocked.

To perform UDP port scanning, we use -sU flag.

```
sudo nmap -sU IP
```

The main challenges with UDP scanning are:

- It's harder to determine if a port is open vs filtered, since open ports often don't respond.
- It's slower than TCP scanning, since retransmissions are needed for each port.
- Many hosts rate-limit ICMP error messages, slowing down the scan

---

SCTP Scanning

So now we will look into SCTP Scanning.

But first, we will start with the basic question - **What is SCTP ?**

SCTP or Stream Control Transmission Protocol is a computer networking protocol that allows two devices to reliably send and receive multiple streams of data at the same time over an internet connection. It is generally used in VOIP services and is considered as an alternative to TCP and UDP protocols.

It works on a 4 way handshake mode, as we can see in the picture.

Now SCTP uses two types of scanning techniques. One is SCTP INIT Scanning and other one is SCTP COOKIE ECHO Scanning. Lets look into each one of them, one by one.

SCTP INIT Scanning

First we have SCTP INIT Scanning. Lets see how it works.

- The scanner sends an SCTP INIT packet to each target port, which is the first step in establishing an SCTP association.
- If the port is open, the target responds with an SCTP INIT-ACK packet, indicating it's ready to establish a connection.
- If the port is closed, the target responds with an SCTP ABORT packet, refusing the connection attempt.
- If no response is received or if an ICMP unreachable error is received, the port is marked as filtered

To perform this scan, we will use the -sY flag.

```
sudo nmap -sY 192.168.29.141
```

As expected we did not find any successful result back because no SCTP based application running on the target.

So in essence, the SCTP scanner is "knocking" on each port to see if it answers back with an INIT-ACK, indicating it's open and listening for SCTP traffic. A closed port responds with an ABORT, while a filtered port doesn't respond at all.

The main advantage of SCTP scanning is that it can find open SCTP ports that would be missed by traditional TCP or UDP scans. SCTP is used by some services like SS7/SIGTRAN, so scanning for it can uncover additional attack surface.

However, SCTP scanning is less common than TCP or UDP scanning, as SCTP is not as widely deployed as the other protocols. It also requires special support in the port scanning tool, which may not always be available.

SCTP Cookie ECHO Scanning

SCTP Cookie ECHO scanning is a technique used to determine if a port on a target system is open or closed, without fully establishing an SCTP association.

Here's how it works:

1. The scanner sends an SCTP packet containing a COOKIE ECHO chunk to the target port

2. If the port is closed, the target will respond with an ABORT chunk, refusing the connection attempt
3. If the port is open, the target will silently drop the COOKIE ECHO packet without sending any response

So in essence, the scanner is "knocking" on the port with a COOKIE ECHO chunk. If it gets an ABORT back, the port is closed. If it gets no response at all, the port is likely open.

To perform this scan, we will use the -sZ flag.

```
sudo nmap -sZ IP
```

Okay, so this time, we do get some output. Unlike the init scan. However the state of the ports are still filtered.

So in essence, the scanner is "knocking" on the port with a COOKIE ECHO chunk. If it gets an ABORT back, the port is closed. If it gets no response at all, the port is likely open.

The main advantages of SCTP Cookie ECHO scanning are:

- It's more stealthy than a regular SCTP INIT scan, making it harder to detect
- It may bypass non-stateful firewalls that block INIT chunks but not COOKIE ECHO chunks.

However, it also has some drawbacks:

- It cannot reliably differentiate between open and filtered ports, often showing both as "open|filtered"
- Advanced IDS/IPS systems may still be able to detect and block SCTP Cookie ECHO scans

---

SSDP Scanning

SSDP stands for Simple Service Discovery Protocol. It's a way for devices on a network to find and talk to each other without needing to be manually configured. Here's how it works:

Imagine you have a lot of devices in your home, like a TV, a computer, and a printer. They all need to be able to talk to each other so you can do things like print a document from your computer or stream a movie from your computer to your TV.

SSDP helps these devices find each other and figure out what they can do for each other. It's like a big "hello, what can you do?" message that each device sends out to the others on the network.

When a device receives this message, it responds with a list of what it can do. For example, the printer might say "I can print documents!" and the TV might say "I can play videos!"

This way, all the devices on the network know what each other can do, and they can work together to make things happen. It's like a big team effort!

But SSDP can also be used for bad things, like helping hackers do something called a **"DDoS attack"**. This is when a lot of devices are tricked into sending messages to a single target, like a website, all at the same time. This can make the website very slow or even crash.

Now that we know what SSDP is used for. We can check our target for it, if it is vulnerable to UPnP exploits.

We will be using Metasploit ssdp_msearch for this, instead of nmap this time.

```
msfconsole

search ssdp_msearch

use auxiliary/scanner/upnp/ssdp_msearch

set RHOSTS 192.168.29.141

run
```

IPV6 Scanning

- At last, we will look into IPV6 Scanning.

IPv6 scanning refers to the process of probing IPv6 networks to discover active hosts, open ports, and running services

1. **Some common IPv6 scanning techniques include:

- Sending ICMPv6 echo requests (ping) to discover active hosts
- Sending TCP SYN packets to common ports to find open ports
- Sending UDP packets to well-known ports to detect services
- Leveraging DNS data, routing information, and other sources to guess likely active addresses

To perform this scan, we will take the scanme.nmap.org as a target and will use -6 flag for the scanning.

```
sudo nmap -6 scanme.nmap.org
```

Before concluding this section, lets discuss one more thing.

As you have noticed, there were multiple ports states that we had encountered till now in our various scanning sessions. Let's understand each, so that you won't get confused with the port status output later in your engagements as this might hinder your exploitation methodology.

- **Open**: This means an application is actively accepting connections on this port. For example, if a web server is running on port 80, it would show as open. Open ports are the main targets for hackers like us who are looking for vulnerabilities.
- **Closed**: The port is accessible and receives the scan packets, but no application is listening on it. Closed ports are still useful for determining if a host is online and for OS fingerprinting.
- **Filtered**: Nmap cannot determine if the port is open because a firewall, router, or host-based software is blocking or filtering the scan packets. These ports provide very little information to the attackers.
- **Unfiltered**: The port is accessible, but Nmap is unable to determine if it is open or closed. This state is only used by the ACK scan, which is used to map firewall rulesets.
- **Open|Filtered**: Nmap is unable to determine if the port is open or filtered. This can happen when open ports don't respond to the scan, or if a packet filter drops the probe and any response.
- **Closed|Filtered**: Nmap is unable to determine if the port is closed or filtered. This state is only used for the IP ID idle scan.

Some of the little but important takeaways from this section will be the use of sudo. In my opinion, we should always run nmap as sudo as it then give us access to the raw TCP/IP implementation of the system and thus helps us in our scans better. The stealth scan is also a default in sudo and just makes the overall process more streamlined.
