**MAJOR PROJECT**

**Title:** Understanding Nmap scanning with Bash script.

**Mentor**

Dr. Priya Sajan

Sr. Project Engineer, CDAC, TVM.

**Team Members & PRN.**

Gopi Aghav (230360940011)

Harshit Upadhyaya (230360940012)

Himanshu Shekhar (230360940013)

Jyothilaxmi P. (230360940014)

Kailas Kalyane (230360940015)

Vipul Bhirud (230360940047)

Contents

[1. Abstract 4](#_Toc144622769)

[2. Introduction 5](#_Toc144622770)

[2.1 What is Scanning? 5](#_Toc144622771)

[2.2 What is Network Scanning? 5](#_Toc144622772)

[2.3 What is Nmap? 5](#_Toc144622773)

[2.4 What is Nmap Scanning? 6](#_Toc144622774)

[3. Methodology 6](#_Toc144622775)

[3.1 Different scans performed by Nmap 6](#_Toc144622776)

[3.2 Ping scan 7](#_Toc144622777)

[3.2.1 -sn (No port scan) 7](#_Toc144622778)

[3.2.2 -Pn (No ping) 8](#_Toc144622779)

[3.2.3 -PU<portlist>(UDPping) 8](#_Toc144622780)

[3.2.4 -PA <port list> (TCP ACK ping) 8](#_Toc144622783)

[3.2.5 -PS <port list> (TCP SYN Ping) 9](#_Toc144622784)

[3.3 ARP Scan (-PR) 9](#_Toc144622785)

[3.4 OS and Version Scan 11](#_Toc144622786)

[3.5 TCP, UDP and SYN Scan 15](#_Toc144622787)

[Transmission Control Protocol (TCP): 15](#_Toc144622788)

[Packet format: 15](#_Toc144622789)

[Step 1: Establish a connection: 16](#_Toc144622790)

[Step 2: Send packets of data: 17](#_Toc144622791)

[Step 3: Close the connection: 18](#_Toc144622792)

[Detecting lost packets: 18](#_Toc144622793)

[Handling out-of-order packets: 19](#_Toc144622794)

[3.5.1 TCP Connect Scans (-sT) 20](#_Toc144622795)

[User Datagram Protocol 21](#_Toc144622796)

[Packet format: 21](#_Toc144622797)

[3.5.2 UDP Scans (-sU) 22](#_Toc144622798)

[3.5.3 SYN SCAN (-sS): 23](#_Toc144622799)

[3.6 Idle Scan: 24](#_Toc144622800)

[3.6.1 Idle Scan Step by Step: 24](#_Toc144622801)

[3.6.2 Finding a Working Idle Scan Zombie Host: 27](#_Toc144622802)

[3.6.3 Executing an Idle Scan 28](#_Toc144622803)

[⸭ DIAGRAM OF IDLE SCAN:- 29](#_Toc144622804)

[3.7 Special Scans 30](#_Toc144622805)

[3.7.1. TCP Null Scan (-sN) 30](#_Toc144622806)

[3.7.2 TCP Fin Scan (-sF) 30](#_Toc144622807)

[3.7.3 TCP Xmas Scans (-sX) 31](#_Toc144622808)

[3.8 Timing Scan 31](#_Toc144622809)

[3.8.1 Nmap Paranoid (-T0) Scan 31](#_Toc144622810)

[3.8.2 Nmap Sneaky (-T1) Scan: 31](#_Toc144622811)

[3.8.3 Nmap Polite (-T2) Scan 32](#_Toc144622812)

[3.8.4 Nmap Normal (-T3) Scan: 32](#_Toc144622813)

[3.8.5 Nmap Aggressive (-T4) Scan: 33](#_Toc144622814)

[3.8.6 Nmap Insane (-T5) Scan: 33](#_Toc144622815)

[3.9 Firewall evasion: 34](#_Toc144622816)

[3.9.1 WHAT IS A FIREWALL? 34](#_Toc144622817)

[3.9.2 Why Firewall? 35](#_Toc144622818)

[3.9.3 How does a firewall work? 36](#_Toc144622822)

[3.9.4 Firewall Evasion using Nmap 37](#_Toc144622824)

[Fragmentation 37](#_Toc144622825)

[MTU---Maximum Transmission Unit 37](#_Toc144622826)

[Data Length 37](#_Toc144622827)

[Decoy scan 38](#_Toc144622828)

[Spoof source address. 38](#_Toc144622829)

[3.10 Nmap Script Engine (NSE) 39](#_Toc144622830)

[4. Scope and Objectives 39](#_Toc144622831)

[5. Figures 39](#_Toc144622832)

[6. References 42](#_Toc144622833)

# Abstract

Quoting from Nmap.org, “Nmap is free, the only barrier to port scanning mastery is knowledge”. Our aim here is to provide knowledge to new users or beginners in understanding various switches of Nmap along with its Nmap scripting engine (NSE).

“ScanSploit” will help beginners with the understanding of these switches with description about switches or NSE scripts which will help in understanding what will be Nmap out.

Keyword: Scanning, Nmap, Nmap NSE scripts.

# Introduction

## What is Scanning?

Scanning, in the context of computer networks and security, refers to the process of systematically exploring a network, device, or system to gather information about its characteristics, vulnerabilities, and potential security risks. Scanning is an essential practice in network administration, cybersecurity, and ethical hacking for various purposes, including network discovery, security assessments, and troubleshooting.

## What is Network Scanning?

Network Scanning refers to the process of systematically examining a network to identify active devices, open ports, services, and potential vulnerabilities. It's a crucial practice in network security and administration that helps administrators and security professionals understand the layout, security posture, and potential risks within a network.

Network Scanning involves sending specific packets of data to target systems or devices and analyzing the responses to gather information about their characteristics. This information can include details about the operating systems, services, and applications running on those systems.

## What is Nmap?

Nmap, short for "Network Mapper," is a powerful open-source network scanning tool used for network discovery, security auditing, and vulnerability assessment. It's widely regarded as one of the most versatile and comprehensive tools for network exploration and analysis. Nmap is available for various platforms and can be run from the command line or through graphical user interfaces (GUIs).

Nmap is often used by network administrators, security professionals, penetration testers, and ethical hackers to assess network security, identify potential vulnerabilities, and gather information about networked devices. It's important to note that Nmap should be used responsibly and within authorized environments. Unauthorized scanning of networks or systems can lead to legal consequences and ethical dilemmas.

## What is Nmap Scanning?

Nmap scanning, also known as network scanning using the Nmap tool, refers to the practice of using the Nmap (Network Mapper) software to systematically discover, analyze, and gather information about devices, hosts, and services within a network. Nmap scanning involves sending carefully crafted packets of data to target systems and analyzing the responses to extract valuable insights about the network's configuration, security posture, and potential vulnerabilities.

Nmap scanning holds significant importance in the realm of network security, administration, and assessment due to its wide range of capabilities and benefits.

Nmap scanning serves multiple purposes, including network reconnaissance, security audits, penetration testing, and troubleshooting. It helps administrators and security professionals to gain insights into their network infrastructure, identify potential security risks, and take proactive measures to enhance security.

Ultimately, Nmap scanning empowers organizations to maintain the security, integrity, and performance of their networks. However, it's essential to use Nmap responsibly and within authorized environments to avoid any negative impact on systems or networks.

# Methodology

## Different scans performed by Nmap

1. Ping Scan
2. ARP Scan
3. OS and Version Scan,
4. TCP and UDP scans, Syn Scan
5. Null Scan, Fin Scan, XMas Scan, ACK Scan
6. Idle Scan
7. Timing Scan
8. Firewall evasion.

## Ping scan

Nmap offers a wide variety of options for customizing the techniques used. Host discovery is sometimes called ping scan, but it goes well beyond the simple ICMP echo request packets associated with the ubiquitous ping tool. Users can skip the discovery step entirely with a list scan (-sL) or by disabling host discovery (-Pn), or engage the network with arbitrary combinations of multi-port TCP SYN/ACK, UDP, SCTP INIT, and ICMP probes. The goal of these probes is to solicit responses that demonstrate that an IP address is actually active (is being used by a host or network device). On many networks, only a small percentage of IP addresses are active at any given time. This is particularly common with private address space such as 10.0.0.0/8. That network has 16 million IPs, but I have seen it used by companies with less than a thousand machines. Host discovery can find those machines in a sparsely allocated sea of IP addresses.

If no host discovery options are given, Nmap sends an ICMP echo request, a TCP SYN packet to port 443, a TCP ACK packet to port 80, and an ICMP timestamp request. (For IPv6, the ICMP timestamp request is omitted because it is not part of ICMPv6.) These defaults are equivalent to the -PE -PS443 -PA80 -PP options. The exceptions to this are the ARP (for IPv4) and Neighbor Discovery (for IPv6) scans which are used for any targets on a local ethernet network. For unprivileged Unix shell users, the default probes are a SYN packet to ports 80 and 443 using the connect system call. This host discovery is often sufficient when scanning local networks, but a more comprehensive set of discovery probes is recommended for security auditing.

### -sn (No port scan)

A no port scan prevents Nmap from doing a port scan after host discovery. It is specified with the -sn option. Used by itself, it is also known as a “ping” scan. It is less intrusive and attracts less attention than a full port scan.

Example:-nmap -sn scanme.nmap.org



### -Pn (No ping)

A no ping scan prevents Nmap from doing any host discovery. It is specified with the -Pn option. By default, Nmap only probes active machines found via host discovery. With this option selected, Nmap will do the requested scanning functions against every target IP specified, as if every one is active. This is done when it is important to find every possible active machine, including ones that may not be responsive to host host discovery.

## -PU<portlist>(UDPping)

## Another host discovery is the udp ping which sends a udp packet to the given ports.for most ports,the packet will be empty though some use a specific payload that is more likely to elicit a response.the payloads are the same used in service and version detection and are defined in nmapviceprobes file.packet content can also be affected with-data,--data-string,and--data-length options.

# Example nmap -PU scanme.nmap.org

## -PA <port list> (TCP ACK ping)

The tcp ack ping is quite similar to the just discussed syn ping.the difference as you could likely guess,is that the tcp ack flag is set instead of the syn flag.such an ack packet purports to be acknowledging data over an established tcp connection,but no such extension exist.so remote host should always respond with RST packet,disclosing their extension in the process.

Example:-nmap -PA scanme.nmap.org

## -PS <port list> (TCP SYN Ping)

This option sends an empty TCP packet with the SYN flag set. The default destination port is 80 (configurable at compiletime by changing DEFAULT\_TCP\_PROBE\_PORT\_SPEC in nmap.h). Alternate ports can be specified as a parameter. The syntax is the same as for the -p except that port type specifiers like T: are not allowed. Examples are -PS22 and -PS2225,80,113,1050,35000. Note that there can be no space between -PS and the port list. If multiple probes specified they will be sent in parrallel.

Example:- nmap -ps scanme.nmap.org

## ARP Scan (-PR)

One of the most common Nmap usage scenarios is to scan an ethernet LAN. On most LANs, especially those using private address ranges granted by [RFC 1918](http://www.rfc-editor.org/rfc/rfc1918.txt), the vast majority of IP addresses are unused at any given time. When Nmap tries to send a raw IP packet such as an ICMP echo request, the operating system must determine the destination hardware (ARP) address corresponding to the target IP so that it can address the ethernet frame properly. This requires it to issue a series of ARP requests. This is shown in [Example 3.11](https://nmap.org/book/host-discovery-techniques.html#host-discovery-ex-eth-ip-scan), where a ping scan is attempted against a local ethernet host. The --send-ip option tells Nmap to send IP level packets (rather than raw ethernet) even though it is a local network. Wireshark output of the three ARP requests and their timing has been pasted into the session.

Example 3.11. Raw IP ping scan of an offline target

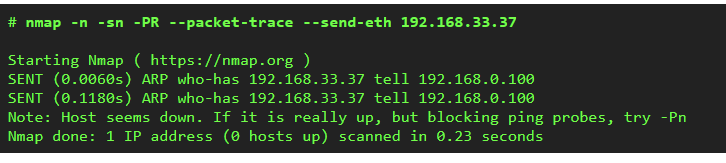
****

This example took more than two seconds to finish because the (Linux) OS sent three ARP requests, one second apart, before giving up on the host. Given that ARP replies usually come within a couple milliseconds, multi-second waits are excessive. Decreasing this timeout period is no priority for OS vendors because the vast majority of packets are sent to hosts that actually exist. Nmap, on the other hand, must send packets to 16 million IPs when given a target such as 10.0.0.0/8. A two second wait for each becomes a huge delay even though many targets are pinged in parallel.

There is another problem with raw IP ping scans on LANs. When a destination host is found to be unresponsive as in the previous example, the source host generally adds an incomplete entry for that destination IP in its kernel ARP table. ARP table space is finite, and some operating systems react badly when it fills up. When Nmap is used in raw IP mode (--send-ip), Nmap sometimes has to wait several minutes for ARP cache entries to expire before it can continue with host discovery.

ARP scanning resolves both problems by putting Nmap in control. Nmap issues the raw ARP requests and handles retransmission and timeout periods at its own discretion. The system ARP cache is bypassed. [Example 3.12](https://nmap.org/book/host-discovery-techniques.html#host-discovery-ex-eth-arp-scan) shows the difference. This ARP scan takes just over a tenth of the time taken by its IP equivalent.

Example 3.12. ARP ping scan of an offline target



In [Example 3.12](https://nmap.org/book/host-discovery-techniques.html#host-discovery-ex-eth-arp-scan), neither the -PR or --send-eth options have any effect. This is because ARP is the default scan type when scanning ethernet hosts that Nmap detects are on a local ethernet network. This includes traditional wired ethernet as well as 802.11 wireless networks. Not only is ARP scanning more efficient as discussed above, it is also more accurate. Hosts frequently block IP-based ping packets, but they generally cannot block ARP requests or responses and still communicate on the network. Even if different ping types (such as -PE or -PS) are specified, Nmap uses ARP instead for any of the targets which are on the same LAN. If you absolutely don't want to do an ARP scan, specify --send-ip as shown in [Example 3.11, “Raw IP ping scan of an offline target”](https://nmap.org/book/host-discovery-techniques.html#host-discovery-ex-eth-ip-scan).

Giving Nmap control to send raw ethernet frames also allows Nmap to control the source MAC address. If you have the only PowerBook in the room at a security conference and a massive ARP scan is initiated from a MAC address registered to Apple, heads may turn in your direction. You can spoof your MAC address with the --spoof-mac option, as discussed in [the section called “MAC Address Spoofing”](https://nmap.org/book/firewall-subversion.html#defeating-firewalls-mac-spoofing).

## OS and Version Scan

[**NMAP**](https://www.geeksforgeeks.org/nmap-command-in-linux-with-examples/)stands for Network Mapper which is an open-source tool used for network exploration and security auditing, in comparison to this, a tool named **Nessus**is used by industry professionals. These tools are mainly used by cybersecurity experts and hackers.

Its main purpose is:

* Provide the list of the live host.
* Find the open Ports.
* The real-time information of a network.
* OS and Port scanning.

The hackers and the cybersecurity expert need to know the Operating System of the machine. It becomes very easy to access a system if we can know the specific open ports or the security holes of the system.[**Network Mapper(NMAP)**](https://www.geeksforgeeks.org/nmap-command-in-linux-with-examples/)NMAP has a database that helps in [**Operating systems (OS)**](https://www.geeksforgeeks.org/operating-systems/)but it is not automatically updated. The database to detect an OS is located at ‘/usr/share/nmap/nmap-os-db’.

[**Operating System(OS)**](https://www.geeksforgeeks.org/operating-systems/)detection is a very long and hectic process. So, before we get our hands dirty we should know about the five separate probes being performed to determine the OS. This probe may consist of one or more packets. The response to each packet (which is sent by the probe) by the target system helps to determine the OS type.

The five different probes are:

* Sequence Generation.
* ICMP Echo.
* TCP Explicit Congestion Notification.
* TCP.
* UDP.

**1. Sequence Generation:**The Sequence Generation Probe consists of six packets that are sent 100 ms apart and are all TCP SYN packets. The result of all these packets will help in [**Operating System(OS)**](https://www.geeksforgeeks.org/operating-systems/)detection.

**2. ICMP Echo:**Two ICMP request packets are sent to the target system with different settings in the packet. The result of all these will help verify the OS type by NMAP.

**3. TCP Explicit Congestion Notification:**Congestion is a slowdown that occurs when a lot of packets are generated and passed by a single router. The packets which are sent are mainly used to get back the responses from the target system. This helps to detect the OS because a specific OS returns a specific value and each OS handles a packet differently.

**4. TCP:**Six packets are sent during this probe, and some packets are sent to open or closed ports with specific packet settings by using the corresponding result we can determine the type of **Operating System(OS)**. The TCP Packets which are sent with varying flags are as follows:

* no flags.
* SYN, FIN, URG, and PSH.
* ACK.
* SYN.
* ACK.
* FIN, PSH, and URG.

**5.**[**UDP**](https://www.geeksforgeeks.org/user-datagram-protocol-udp/)**:**UDP probe consists of a single packet that is sent to a closed port. If the port used on the target system is closed and an ICMP Port Unreachable message is returned it specifies that there is no Firewall.

**OS detection using NMAP**

Now we need to run the actual commands to perform OS detection using NMAP, and at first, we will get the IP address of the host system, and then will perform a scan to get all active devices on the network.

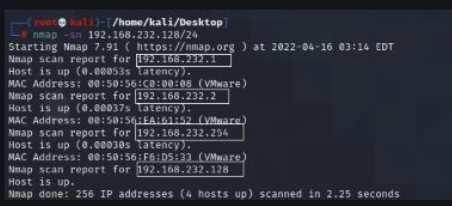
**Step 1:**Getting the IP of the System

ifconfig



**Step 2:**List of active devices in the Network

nmap -sn 192.168.232.128/24



**Let’s do an SYN scan with OS detection in one of the active IPs**

**Let’s select IP**: 192.168.232.2

nmap -sS 192.168.232.2 -O



**Running:**VMware Player.

**OS details:**VMware Player virtual NAT device.

**Let’s now perform an Aggressive scan To guess the OS**

* **-sV**stands for Service version.
* **-A** stands for Aggressive.

It will only display the chance of Operation System (OS) on the host computer with the help of Probability and Percentage.

nmap -sV 192.168.232.2 -A



## TCP, UDP and SYN Scan

### Transmission Control Protocol (TCP):

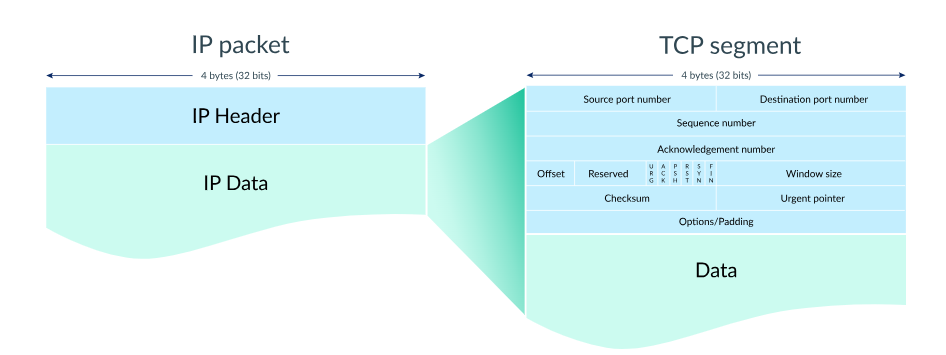
The **Transmission Control Protocol (TCP)** is a transport protocol that is used on top of IP to ensure reliable transmission of packets.

TCP includes mechanisms to solve many of the problems that arise from packet-based messaging, such as lost packets, out-of-order packets, duplicate packets, and corrupted packets.

Since TCP is the protocol used most commonly on top of IP, the Internet protocol stack is sometimes referred to as **TCP/IP**.

### Packet format:

When sending packets using TCP/IP, the data portion of each IP packet is formatted as a **TCP segment**:



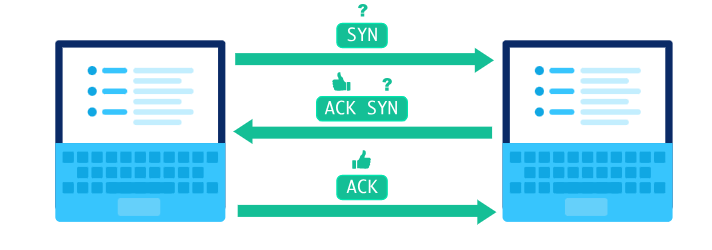
Each TCP segment contains a header and data. The TCP header contains many more fields than the UDP header and can range in size from 202020 to 606060 bytes, depending on the size of the options field.

The TCP header shares some fields with the UDP header: source port number, destination port number, and checksum.

Let's step through the process of transmitting a packet with TCP/IP.

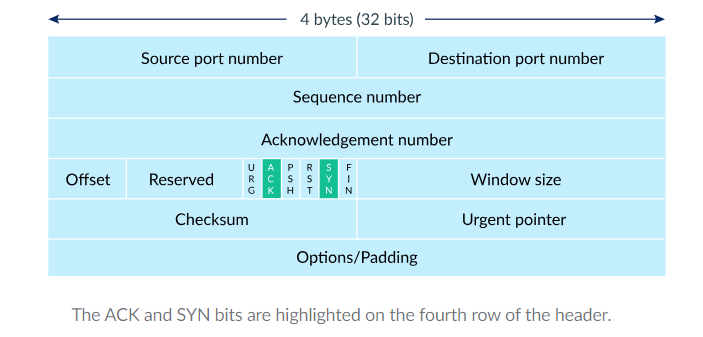
### Step 1: Establish a connection:

When two computers want to send data to each other over TCP, they first need to establish a connection using a **three-way handshake**.



The first computer sends a packet with the SYN bit set to 111 (SYN = "synchronize?"). The second computer sends back a packet with the ACK bit set to 111 (ACK = "acknowledge!") plus the SYN bit set to 111. The first computer replies back with an ACK.

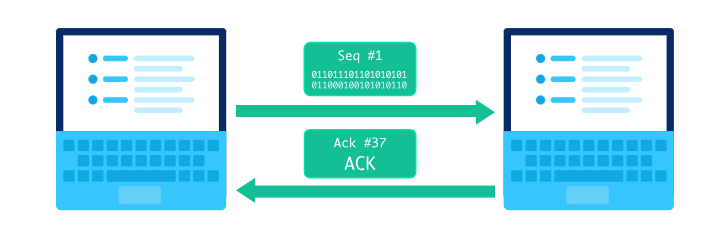
The SYN and ACK bits are both part of the TCP header:



In fact, the three packets involved in the three-way handshake do not typically include any data. Once the computers are done with the handshake, they're ready to receive packets containing actual data.

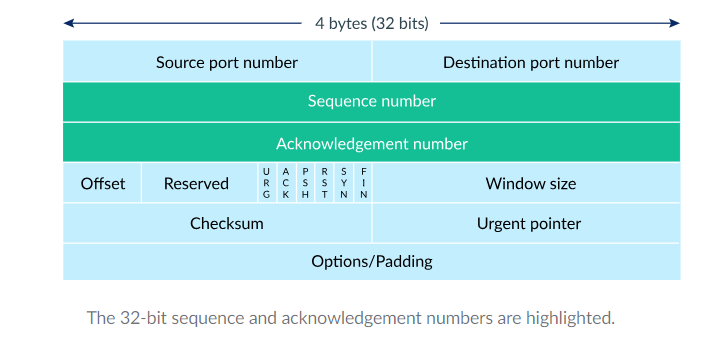
### Step 2: Send packets of data:

When a packet of data is sent over TCP, the recipient must always acknowledge what they received.



The first computer sends a packet with data and a sequence number. The second computer acknowledges it by setting the ACK bit and increasing the acknowledgment number by the length of the received data.

The sequence and acknowledgment numbers are part of the TCP header:



Those two numbers help the computers to keep track of which data was successfully received, which data was lost, and which data was accidentally sent twice.

### Step 3: Close the connection:

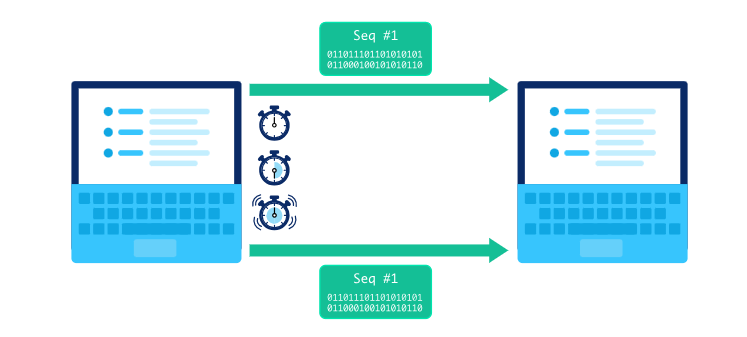
Either computer can close the connection when it no longer wants to send or receive data.



A computer initiates closing the connection by sending a packet with the FIN bit set to 1 (FIN = finish). The other computer replies with an ACK and another FIN. After one more ACK from the initiating computer, the connection is closed.

### Detecting lost packets:

TCP connections can detect lost packets using a timeout.

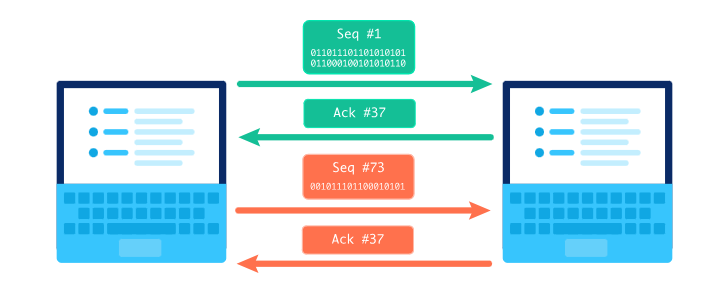


After sending off a packet, the sender starts a timer and puts the packet in a retransmission queue. If the timer runs out and the sender has not yet received an ACK from the recipient, it sends the packet again.

The retransmission may lead to the recipient receiving duplicate packets if a packet was not actually lost but is just very slow to arrive or be acknowledged. If so, the recipient can simply discard duplicate packets. It's better to have the data twice than not at all!

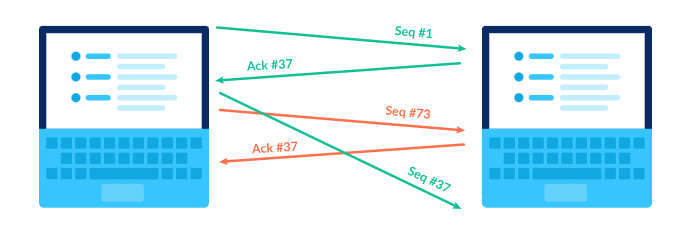
### Handling out-of-order packets:

TCP connections can detect out-of-order packets by using the sequence and acknowledgment numbers.



When the recipient sees a higher sequence number than what they have acknowledged so far, they know that they are missing at least one packet in between. In the situation pictured above, the recipient sees a sequence number of #73 but expects a sequence number of #37. The recipient lets the sender know there's something amiss by sending a packet with an acknowledgment number set to the expected sequence number.

Sometimes the missing packet is simply taking a slower route through the Internet and it arrives soon after.



Other times, the missing packet may actually be a lost packet and the sender must retransmit the packet.



In both situations, the recipient has to deal with out of order packets. Fortunately, the recipient can use the sequence numbers to reassemble the packet data in the correct order.

### 3.5.1 [TCP Connect Scans (-sT)](https://www.digitalocean.com/community/tutorials/nmap-switches-scan-types#1-tcp-connect-scans-st)

In this type of scan, Nmap sends a [TCP packet](https://www.digitalocean.com/community/tutorials/difference-between-tcp-and-udp-protocols) to a port with the SYN flag set. In this scenario two things can occur:

* The target responds with an RST packet that signifies that the port is closed.
* Target doesn’t respond at all, probably due to a firewall dropping all incoming packets in which case the port will be considered filtered
* The target responds back with a TCP packet with the SYN/ACK flags set which would signify that the port is open and then Nmap would respond with a TCP packet with the ACK flag set and hence would complete the TCP 3-way handshake.

This is not a very reliable scan technique as it is easy to configure a firewall rule to respond back with RST packets or drop all incoming packets. Also, this method is extremely slow as it waits for the entire TCP 3-way handshake.

## User Datagram Protocol

A UDP scan is a type of network scanning technique used to identify open UDP (User Datagram Protocol) ports on a target system. Unlike TCP, UDP is a connectionless protocol, meaning it doesn't establish a formal connection before transmitting data. UDP scans can be more challenging than TCP scans because UDP is less predictable and does not guarantee reliable communication or response.

In a UDP scan, the scanning tool sends UDP packets to various UDP ports on the target system and analyses the responses to determine whether the ports are open, closed, or filtered by firewalls.

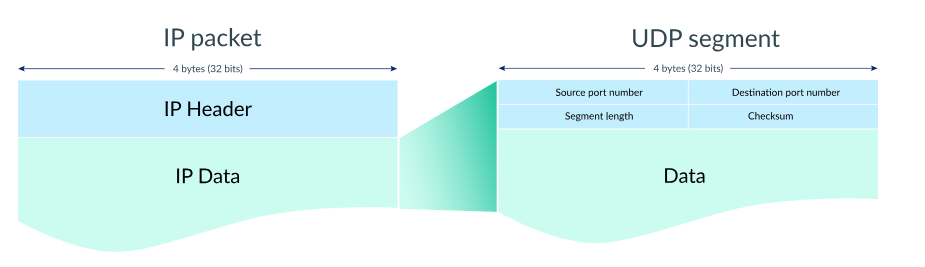
The **User Datagram Protocol (UDP)** is a lightweight data transport protocol that works on top of IP.

UDP provides a mechanism to detect corrupt data in packets, but it does not attempt to solve other problems that arise with packets, such as lost or out-of-order packets. That's why UDP is sometimes known as the **Unreliable Data Protocol**.

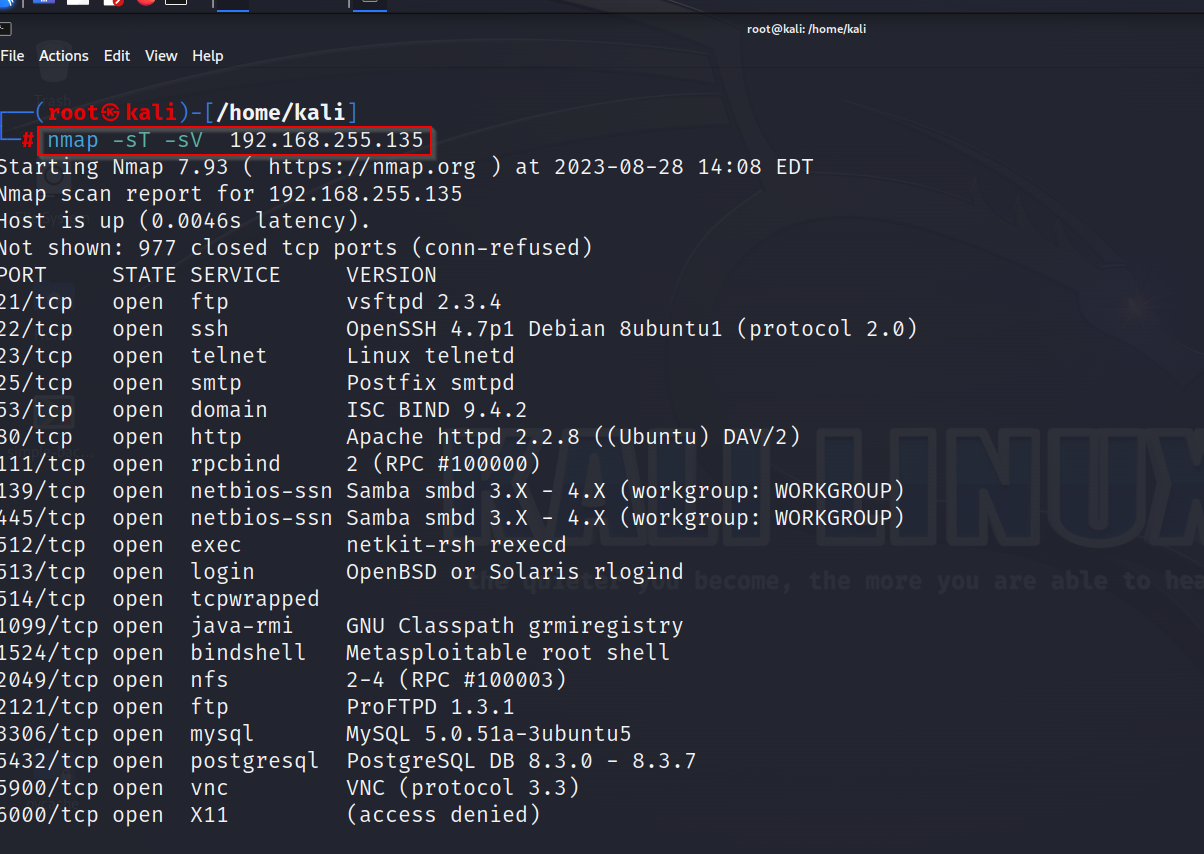
UDP is simple but fast, at least in comparison to other protocols that work over IP. It's often used for time-sensitive applications (such as real-time video streaming) where speed is more important than accuracy.

### Packet format:

When sending packets using UDP over IP, the data portion of each IP Packet is formatted as a **UDP segment**.



Each UDP segment contains an 8-byte header and variable length data.



### 3.5.2 [UDP Scans (-sU)](https://www.digitalocean.com/community/tutorials/nmap-switches-scan-types#3-udp-scans-su)

UDP scans are much less reliable than the previous two as UDP connections are stateless by nature. This means that there’s no “feedback mechanism” like TCP. UDP works on the principle “Fire and Forget” which means that it sends packets directed to targets at certain ports and hopes that they would make it. This gives more emphasis on speed than quality. However, the lack of a feedback mechanism makes it difficult to identify open ports.

When a UDP packet is sent to a target port, there might be three scenarios:

* Usually there is no response received in which case Nmap marks the port as open/filtered. If no response is received yet, it sends another UDP packet to double check and if no response is received, it marks the port as open/filtered and moves on
* It might get a UDP response back which is very rare. In such a scenario, the port is marked open
* If the port is closed and it receives an ICMP echo request back it signifies that the port is unreachable.

### 3.5.3 SYN SCAN (-sS):

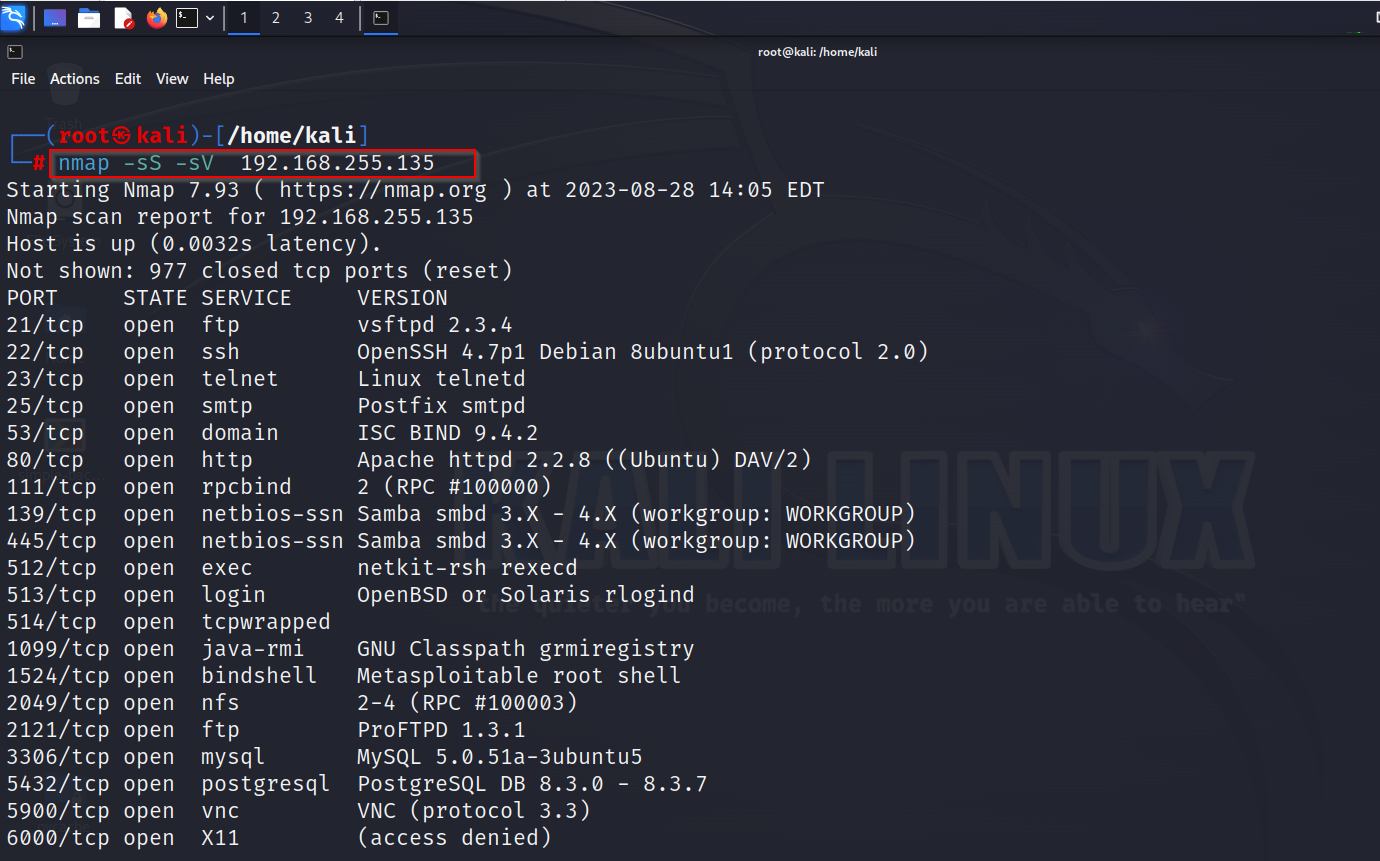
SYN scans, also known as “Half-Open” or “Stealth Scan” are an improvement over the previous method. In the previous method where we were sending back a TCP packet with the ACK flag set after receiving an SYN/ACK packet, now we would be sending an RST packet. This prevents the server from repeatedly trying to make the requests and massively reduces scan times.

This method is an improvement on the previous ones because:

* They are faster
* They might be able to bypass some primitive firewalls
* Often, SYN Scans are not logged by applications running on the ports as most applications start logging a connection only after it has been fully established which is not the case with SYN Scans

However, it is not advisable to run SYN Scans on production environments as it might break certain unstable applications. It is also to be noted that SYN scans also require [sudo privileges](https://www.digitalocean.com/community/tutorials/sudo-command-in-linux) because they need to craft raw packets.

In fact, when run with sudo privileges, Nmap defaults to SYN Scans, otherwise, it defaults to TCP scan.



## 3.6 Idle Scan:

An **idle scan** is a [TCP](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) [port scan](https://en.wikipedia.org/wiki/Port_scanner) method for determining what services are open on a target [computer](https://en.wikipedia.org/wiki/Computer)[[1]](https://en.wikipedia.org/wiki/Idle_scan#cite_note-ERIK77-1) without leaving traces pointing back at oneself. This is accomplished by using [packet spoofing](https://en.wikipedia.org/wiki/IP_address_spoofing) to impersonate another computer (called a "*zombie*") so that the target believes it's being accessed by the zombie. The target will respond in different ways depending on whether the port is open, which can in turn be detected by querying the zombie.

In 1998, security researcher Antirez (who also wrote the hping2 tool used in parts of this book) posted to the Bugtraq mailing list an ingenious new port scanning technique. Idle scan, as it has become known, allows for completely blind port scanning. Attackers can actually scan a target without sending a single packet to the target from their own IP address! Instead, a clever side-channel attack allows for the scan to be bounced off a dumb “zombie host”. Intrusion detection system (IDS) reports will finger the innocent zombie as the attacker. Besides being extraordinarily stealthy, this scan type permits discovery of IP-based trust relationships between machines.

While idle scanning is more complex than any of the techniques discussed so far, you don't need to be a TCP/IP expert to understand it. It can be put together from these basic facts:

* One way to determine whether a TCP port is open is to send a SYN (session establishment) packet to the port. The target machine will respond with a SYN/ACK (session request acknowledgment) packet if the port is open, and RST (reset) if the port is closed. This is the basis of the previously discussed SYN scan.
* A machine that receives an unsolicited SYN/ACK packet will respond with a RST. An unsolicited RST will be ignored.
* Every IP packet on the Internet has a fragment identification number (IP ID). Since many operating systems simply increment this number for each packet they send, probing for the IPID can tell an attacker how many packets have been sent since the last probe.

By combining these traits, it is possible to scan a target network while forging your identity so that it looks like an innocent zombie machine did the scanning.

### 3.6.1 Idle Scan Step by Step:

Fundamentally, an idle scan consists of three steps that are repeated for each port:

1. Probe the zombie's IP ID and record it.
2. Forge a SYN packet from the zombie and send it to the desired port on the target. Depending on the port state, the target's reaction may or may not cause the zombie's IP ID to be incremented.
3. Probe the zombie's IP ID again. The target port state is then determined by comparing this new IP ID with the one recorded in step 1.

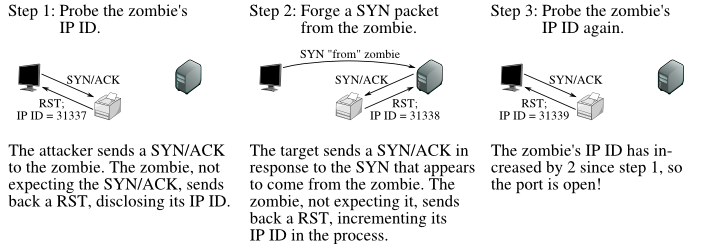
After this process, the zombie's IP ID should have increased by either one or two. An increase of one indicates that the zombie hasn't sent out any packets, except for its reply to the attacker's probe. This lack of sent packets means that the port is not open (the target must have sent the zombie either a RST packet, which was ignored, or nothing at all). An increase of two indicates that the zombie sent out a packet between the two probes. This extra packet usually means that the port is open (the target presumably sent the zombie a SYN/ACK packet in response to the forged SYN, which induced a RST packet from the zombie). Increases larger than two usually signify a bad zombie host. It might not have predictable IP ID numbers, or might be engaged in communication unrelated to the idle scan.

Even though what happens with a closed port is slightly different from what happens with a filtered port, the attacker measures the same result in both cases, namely, an IP ID increase of 1. Therefore it is not possible for the idle scan to distinguish between closed and filtered ports. When Nmap records an IP ID increase of 1 it marks the port closed|filtered.

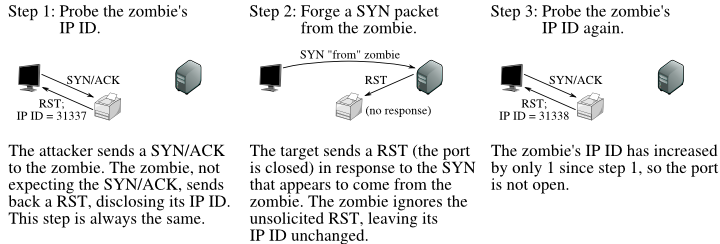
For those wanting more detail, the following three diagrams show exactly what happens in the three cases of an open, closed, and filtered port. The actors in each are:

* the attacker,  the zombie, and  the target.

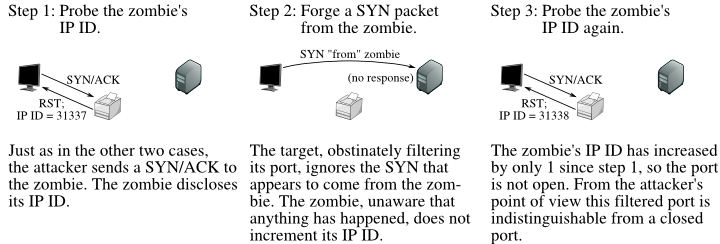
**Figure - Idle scan of an open port:**



**Figure - Idle scan of a closed port:**



**Figure - Idle scan of a filtered port:**



Idle scan is the ultimate stealth scan. Nmap offers decoy scanning (-D) to help users shield their identity, but that (unlike idle scan) still requires an attacker to send some packets to the target from his real IP address in order to get scan results back. One upshot of idle scan is that intrusion detection systems will generally send alerts claiming that the zombie machine has launched a scan against them. So it can be used to frame some other party for a scan. Keep this possibility in mind when reading alerts from your IDS.

A unique advantage of idle scan is that it can be used to defeat certain packet filtering firewalls and routers. IP source address filtering is a common (though weak) security mechanism for limiting machines that may connect to a sensitive host or network. For example, a company database server might only allow connections from the public web server that accesses it. Or a home user might only allow SSH (interactive login) connections from his work machines.

A more disturbing scenario occurs when some company bigwig demands that network administrators open a firewall hole so he can access internal network resources from his home IP address. This can happen when executives are unwilling or unable to use secure VPN alternatives.

Idle scanning can sometimes be used to map out these trust relationships. The key factor is that idle scan results list open ports from the zombie host's perspective. A normal scan against the aforementioned database server might show no ports open, but performing an idle scan while using the web server's IP as the zombie could expose the trust relationship by showing the database-related service ports as open.

Mapping out these trust relationships can be very useful to attackers for prioritizing targets. The web server discussed above may seem mundane to an attacker until she notices its special database access.

A disadvantage to idle scanning is that it takes far longer than most other scan types. Despite the optimized algorithms described in [the section called “Idle Scan Implementation Algorithms”](https://nmap.org/book/idlescan.html#scan-methods-idle-scan-algorithms), A 15-second SYN scan could take 15 minutes or more as an idle scan. Another issue is that you must be able to spoof packets as if they are coming from the zombie and have them reach the target machine. Many ISPs (particularly dialup and residential broadband providers) now implement egress filtering to prevent this sort of packet spoofing. Higher end providers (such as colocation and T1 services) are much less likely to do this. If this filtering is in effect, Nmap will print a quick error message for every zombie you try. If changing ISPs is not an option, you might try using another IP on the same ISP network. Sometimes the filtering only blocks spoofing of IP addresses that are *outside* the range used by customers. Another challenge with idle scan is that you must find a working zombie host, as described in the next section.

### 3.6.2 Finding a Working Idle Scan Zombie Host:

The first step in executing an IP ID idle scan is to find an appropriate zombie. It needs to assign IP ID packets incrementally on a global (rather than per-host it communicates with) basis. It should be idle (hence the scan name), as extraneous traffic will bump up its IP ID sequence, confusing the scan logic. The lower the latency between the attacker and the zombie, and between the zombie and the target, the faster the scan will proceed.

When an idle scan is attempted, Nmap tests the proposed zombie and reports any problems with it. If one doesn't work, try another. Enough Internet hosts are vulnerable that zombie candidates aren't hard to find. Since the hosts need to be idle, choosing a well-known host such as www.yahoo.com or google.com will almost never work.

A common approach is to simply execute a Nmap ping scan of some network. You could use Nmap's random IP selection mode (-iR), but that is likely to result in far away zombies with substantial latency. Choosing a network near your source address, or near the target, produces better results. You can try an idle scan using each available host from the ping scan results until you find one that works. As usual, it is best to ask permission before using someone's machines for unexpected purposes such as idle scanning.

We didn't just choose a printer icon to represent a zombie in our illustrations to be funny—simple network devices often make great zombies because they are commonly both underused (idle) and built with simple network stacks which are vulnerable to IP ID traffic detection.

Performing a port scan and OS identification (-O) on the zombie candidate network rather than just a ping scan helps in selecting a good zombie. As long as verbose mode (-v) is enabled, OS detection will usually determine the IP ID sequence generation method and print a line such as “IP ID Sequence Generation: Incremental”. If the type is given as Incremental or Broken little-endian incremental, the machine is a good zombie candidate. That is still no guarantee that it will work, as Solaris and some other systems create a new IP ID sequence for each host they communicate with. The host could also be too busy. OS detection and the open port list can also help in identifying systems that are likely to be idle.

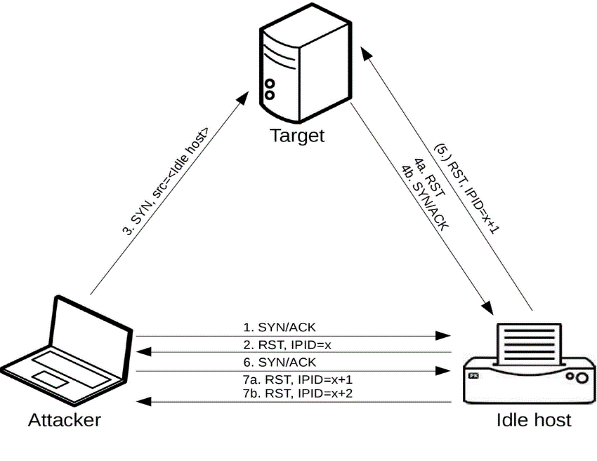
Another approach to identifying zombie candidates is the run the [ipidseq](https://nmap.org/nsedoc/scripts/ipidseq.html" \t "_top) NSE script against a host. This script probes a host to classify its IP ID generation method, then prints the IP ID classification much like the OS detection does. Like most NSE scripts, ipidseq.nse can be run against many hosts in parallel, making it another good choice when scanning entire networks looking for suitable hosts.

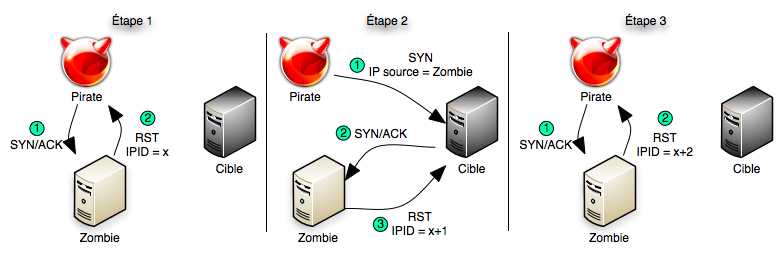
While identifying a suitable zombie takes some initial work, you can keep re-using the good ones.

### 3.6.3 Executing an Idle Scan

Once a suitable zombie has been found, performing a scan is easy. Simply specify the zombie hostname to the -sI option and Nmap does the rest.  shows an example of Ereet scanning the Recording Industry Association of America by bouncing an idle scan off an Adobe machine named Kiosk.

### ⸭ DIAGRAM OF IDLE SCAN:-





**Example - An idle scan against the RIAA:**

# **nmap -Pn -p- -sI kiosk.adobe.com www.riaa.com**

Starting Nmap ( https://nmap.org )

Idlescan using zombie kiosk.adobe.com (192.150.13.111:80); Class: Incremental

Nmap scan report for 208.225.90.120

(The 65522 ports scanned but not shown below are in state: closed)

Port State Service

21/tcp open ftp

25/tcp open smtp

80/tcp open http

111/tcp open sunrpc

135/tcp open loc-srv

443/tcp open https

1027/tcp open IIS

1030/tcp open iad1

2306/tcp open unknown

5631/tcp open pcanywheredata

7937/tcp open unknown

7938/tcp open unknown

36890/tcp open unknown

Nmap done: 1 IP address (1 host up) scanned in 2594.47 seconds

From the scan above, we learn that the RIAA is not very security conscious (note the open PC Anywhere, portmapper, and Legato nsrexec ports). Since they apparently have no firewall, it is unlikely that they have an IDS. But if they do, it will show kiosk.adobe.com as the scan culprit. The -Pn option prevents Nmap from sending an initial ping packet to the RIAA machine. That would have disclosed Ereet's true address. The scan took a long time because -p- was specified to scan all 65K ports. Don't try to use kiosk for your scans, as it has already been removed.

By default, Nmap forges probes to the target from the source port 80 of the zombie. You can choose a different port by appending a colon and port number to the zombie’s name (e.g. -sI kiosk.adobe.com:113). The chosen port must not be filtered from the attacker or the target. A SYN scan of the zombie should show the port in the open or closed state.

## 3.7 Special Scans

Some popular scan types which are even “stealthier” than a TCP SYN scan.

### 3.7.1. TCP Null Scan (-sN)

In TCP Null Scans, the TCP packets sent don’t have any of the flag’s set. According to RFC, under such a circumstance, the target should respond back with an RST if the port is closed.

### 3.7.2 TCP Fin Scan (-sF)

This is very similar to the TCP Null Scan except for the fact that instead of sending a completely empty TCP packet, it sends a packet with the FIN flag set which is used to gracefully close a connection. Accordingly the target must respond back with an RST for closed ports as per RFC.

### 3.7.3 [TCP Xmas Scans (-sX)](https://www.digitalocean.com/community/tutorials/nmap-switches-scan-types#3-tcp-xmas-scans-sx)

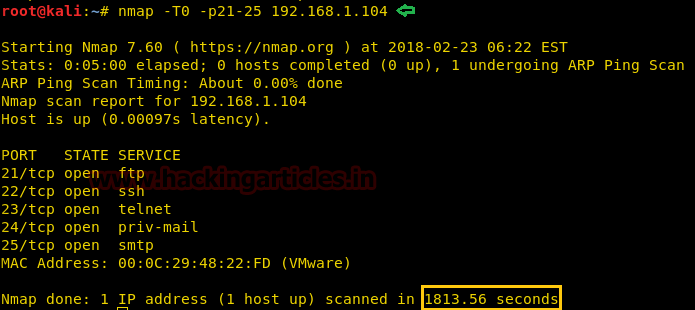
TCP Xmas Scans is also very similar to the last two scan techniques except for the fact that they use TCP packets with the PSH, URG and FIN flags set. Like the last two scan types, this too expects RST packets for closed ports under RFC.

## 3.8 Timing Scan

### **3.8.1 Nmap Paranoid (-T0) Scan**

This template is used for sending packets very slowly as only one port is scanned at a time. The time difference between the two packets sent is 5 minutes.

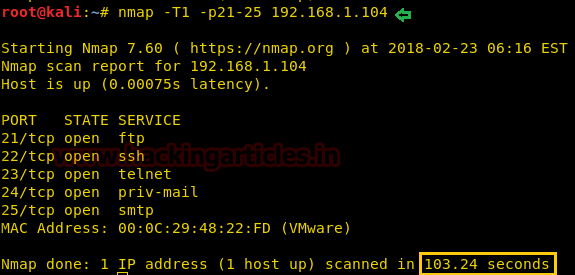
nmap -T0 -p21-25 192.168.1.104



### **3.8.2 Nmap Sneaky (-T1) Scan:**

This template is used for sending packets quickly but still slower than a normal scan. The time difference between the two packets sent is 15 seconds.

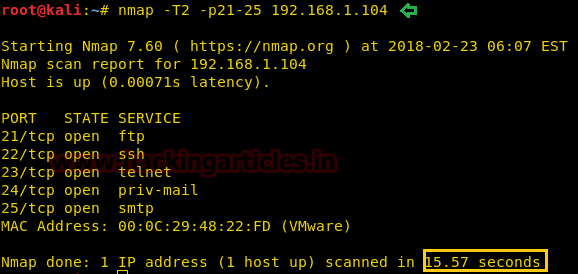
nmap -T1 -p21-25 192.168.1.104



### **3.8.3 Nmap Polite (-T2) Scan**

This template is used for sending packets quickly then –T0 and –T1 but still slower than a normal scan. The time difference between the two packets sent is 0.4 seconds.

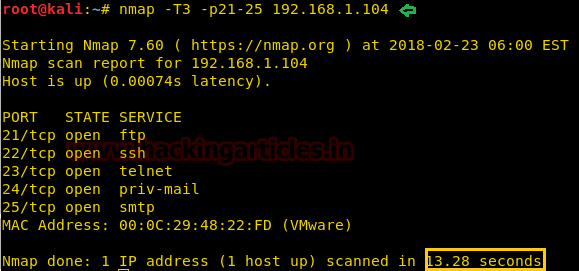
nmap -T2 -p21-25 192.168.1.104



### **3.8.4 Nmap Normal (-T3) Scan:**

This is the default nmap timing template which is used when -T argument is not specified.

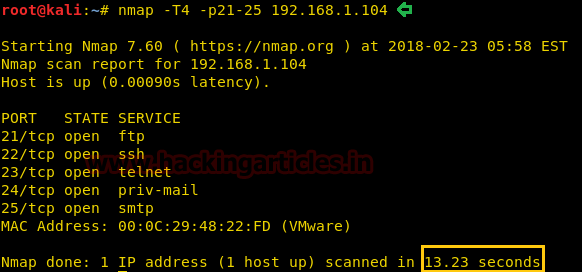
nmap -T3 -p21-25 192.168.1.104



### **3.8.5 Nmap Aggressive (-T4) Scan:**

This template is used for sending packets very fast and **waits only 1.25 seconds** for the response. The time difference between the two packets sent is up to 10 milliseconds. Nmap official documentation recommends using –T4 for “reasonably modern and reliable networks”.

nmap -T4 –p21-25 192.168.1.104

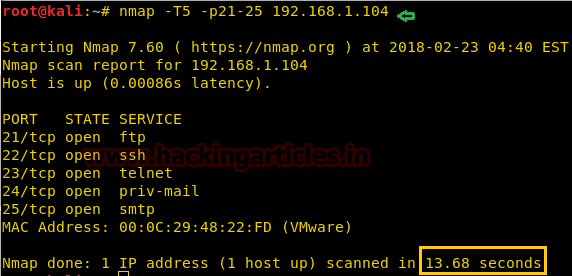


### **3.8.6 Nmap Insane (-T5) Scan:**

This template is used for sending packets insanely fast and **waits only 0.3 seconds** for the response. The time difference between the two packets sent is up to 5 milliseconds. This timing template makes the scan superfast but the accuracy is sacrificed sometimes. Nmap gives-up on a host if it couldn’t complete the scan within 15 minutes. Other than that, -T5 should be used only on a fast network and high-end systems as sending packets this fast can affect the working of the network or system and can result in system failure.

For using timing template use the **attribute –T<0-5>** after Nmap while scanning a target network

nmap -T5 -p21-25 192.168.1.104

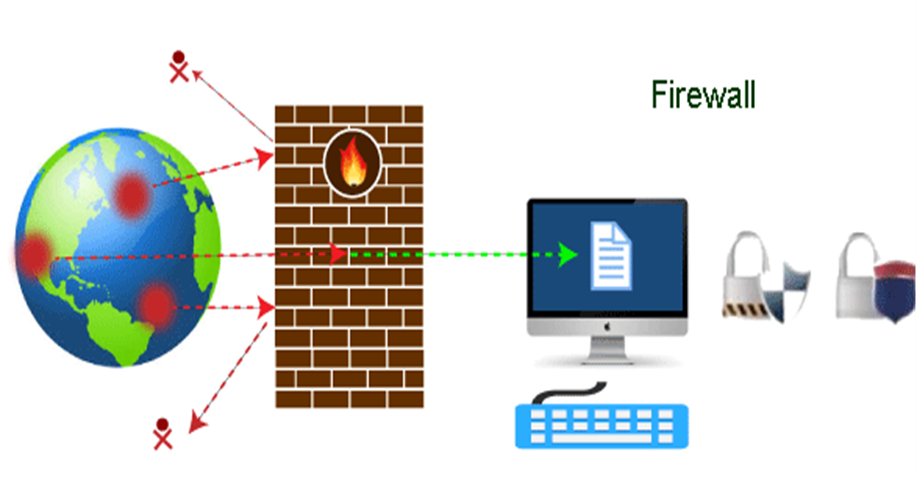


# 3.9 Firewall evasion:

### 3.9.1 WHAT IS A FIREWALL?

A firewall can be defined as a special type of network security device or a software program that monitors and filters incoming and outgoing network traffic based on a defined set of security rules. It acts as a barrier between internal private networks and external sources (such as the public Internet).

The primary purpose of a firewall is to allow non-threatening traffic and prevent malicious or unwanted data traffic for protecting the computer from viruses and attacks. A firewall is a cybersecurity tool that filters network traffic and helps users block malicious software from accessing the [Internet](https://www.javatpoint.com/internet) in infected computers.

****

### 3.9.2 Why Firewall?

Firewalls are primarily used to prevent malware and network-based attacks. Additionally, they can help in blocking application-layer attacks. These firewalls act as a gatekeeper or a barrier. They monitor every attempt between our computer and another network. They do not allow data packets to be transferred through them unless the data is coming or going from a user-specified trusted source.

Firewalls are designed in such a way that they can react quickly to detect and counter-attacks throughout the network. They can work with rules configured to protect the network and perform quick assessments to find any suspicious activity. In short, we can point to the firewall as a traffic controller.

**Some of the important risks of not having a firewall are:**

Open Access:

### If a computer is running without a firewall, it is giving open access to other networks. This means that it is accepting every kind of connection that comes through someone. In this case, it is not possible to detect threats or attacks coming through our network. Without a firewall, we make our devices vulnerable to malicious users and other unwanted sources.

Lost or Comprised Data:

### Without a firewall, we are leaving our devices accessible to everyone. This means that anyone can access our device and have complete control over it, including the network. In this case, cybercriminals can easily delete our data or use our personal information for their benefit.

Network Crashes:

### In the absence of a firewall, anyone could access our network and shut it down. It may lead us to invest our valuable time and money to get our network working again.

Therefore, it is essential to use firewalls and keep our network, computer, and data safe and secure from unwanted sources.

### 3.9.3 How does a firewall work?

A firewall system analyzes network traffic based on pre-defined rules. It then filters the traffic and prevents any such traffic coming from unreliable or suspicious sources. It only allows incoming traffic that is configured to accept.

Typically, firewalls intercept network traffic at a computer's entry point, known as a port. Firewalls perform this task by allowing or blocking specific data packets (units of communication transferred over a digital network) based on pre-defined security rules. Incoming traffic is allowed only through trusted IP addresses, or sources.

## 

### 3.9.4 Firewall Evasion using Nmap

## Fragmentation

* command : nmap –f 192.168.255.135 –p 21
* (To fragment a packet data into fixed 8 bytes size)
* -f (fragment packets); --mtu (using the specified MTU)
* The -f option causes the requested scan (including host discovery scans) to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing.

## MTU---Maximum Transmission Unit

* command : nmap --mtu= 8 192.168.255.135 –p 21
* ( packet size is mentioned as 8 bytes)
* command : nmap --mtu= 16 192.168.255.135 –p 21
* (packet size is mentioned as 16 bytes)

## Data Length

* command : nmap –data-length =25 192.168.255.135 -p 21
* (To append the data length=25 to the syn packet)
* --data-length *<number>* (Append random data to sent packets)
* Normally Nmap sends minimalist packets containing only a header. So its TCP packets are generally 40 bytes and ICMP echo requests are just 28. Some UDP ports and IP protocols get a custom payload by default. This option tells Nmap to append the given number of random bytes to most of the packets it sends, and not to use any protocol-specific payloads. (Use --data-length 0 for no random or protocol-specific payloads. OS detection (-O) packets are not affected because accuracy there requires probe consistency, but most pinging and portscan packets support this. It slows things down a little, but can make a scan slightly less conspicuous.

## Decoy scan

* command : nmap –D RND: 5 192.168.255.135 -p 21
* ( 5 randomly generated ip, It will be difficult to find the legitimate ip from random targets)
* command : nmap –D RND: 5 -f 192.168.255.135 -p 21
* (complex result) -D *<decoy1>*[,*<decoy2>*][,ME][,...] (Cloak a scan with decoys)
* Causes a decoy scan to be performed, which makes it appear to the remote host that the host(s) you specify as decoys are scanning the target network too. Thus their IDS might report 5–10 port scans from unique IP addresses, but they won't know which IP was scanning them and which were innocent decoys. While this can be defeated through router path tracing, response-dropping, and other active mechanisms, it is generally an effective technique for hiding your IP address.

## Spoof source address.

command : -S *<IP\_Address>*

(use -S with the IP address of the interface you wish to send packets through)

* -S *<IP\_Address>* (Spoof source address)
* In some circumstances, Nmap may not be able to determine your source address (Nmap will tell you if this is the case). In this situation, use -S with the IP address of the interface you wish to send packets through.

## 3.10 Nmap Script Engine (NSE)

The Nmap Scripting Engine (NSE) is one of Nmap's most powerful and flexible features. NSE scripts are organized into 14 categories, namely, *auth, brute, broadcast, default, discovery, intrusive, dos, exploit, fuzzer, malware, safe, version, external, vuln*. Of these 14 categories we will be focusing on only 7 which are shown in *Nmap Figure 5.4*.

# Scope and Objectives

Before using the “ScanSploit” tool it is recommended that user should be aware of the scope. Scope and objective of this particular tools is to have basic understanding of the Nmap and its various switches so that one can use them wisely and efficiently in future.

# Figures

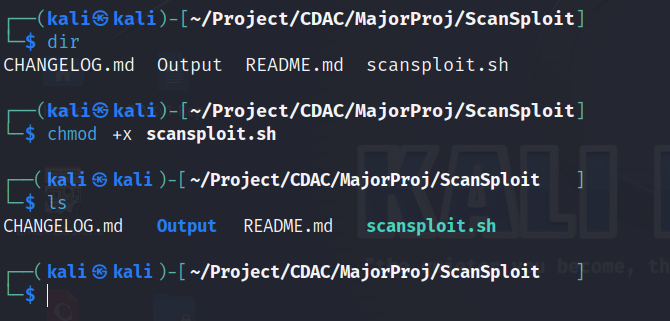
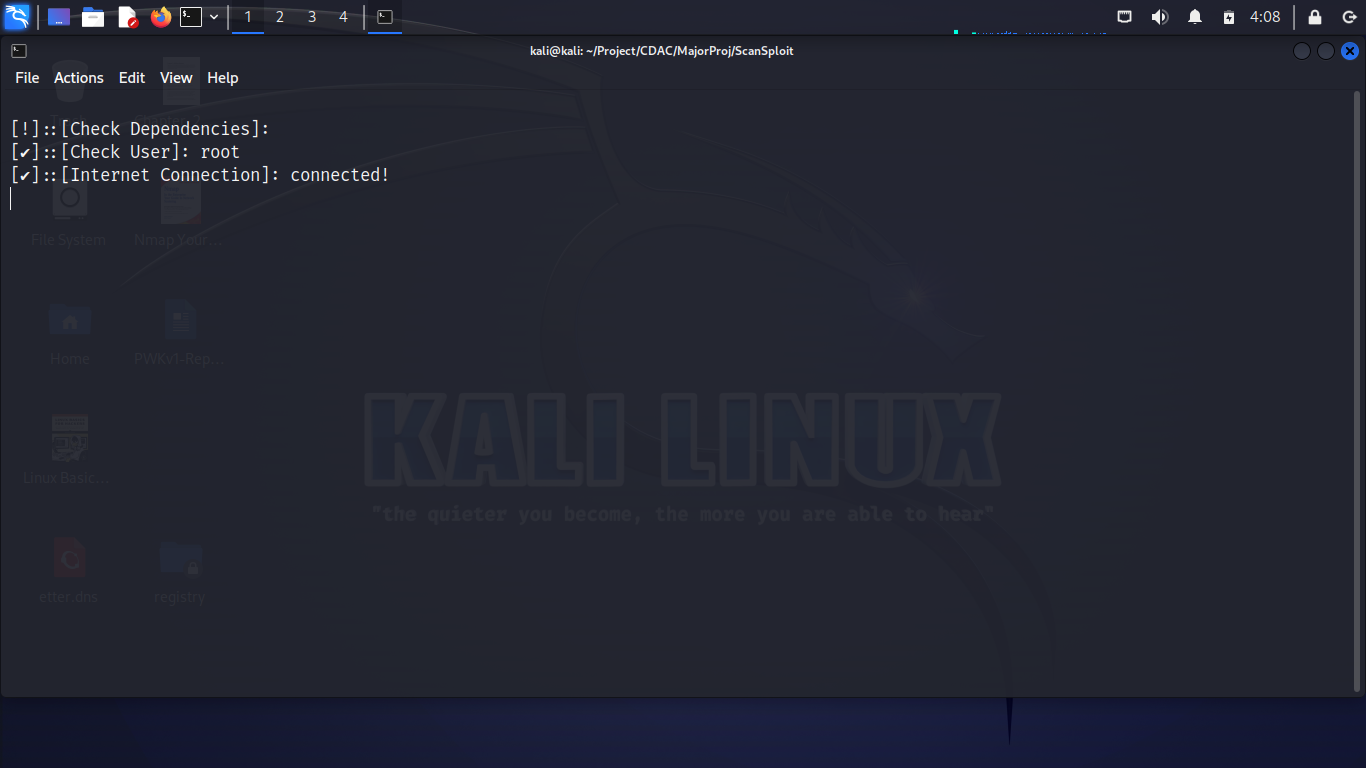


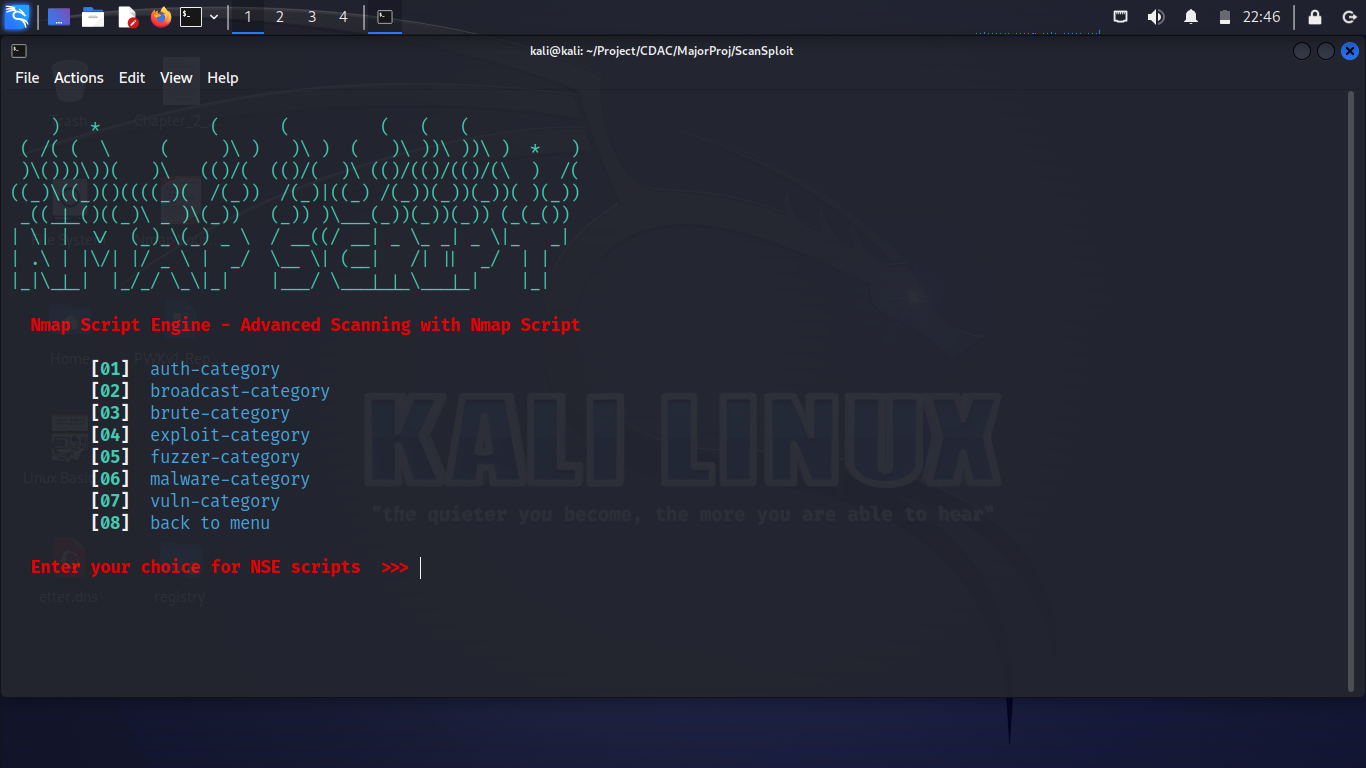
Figure 5.1



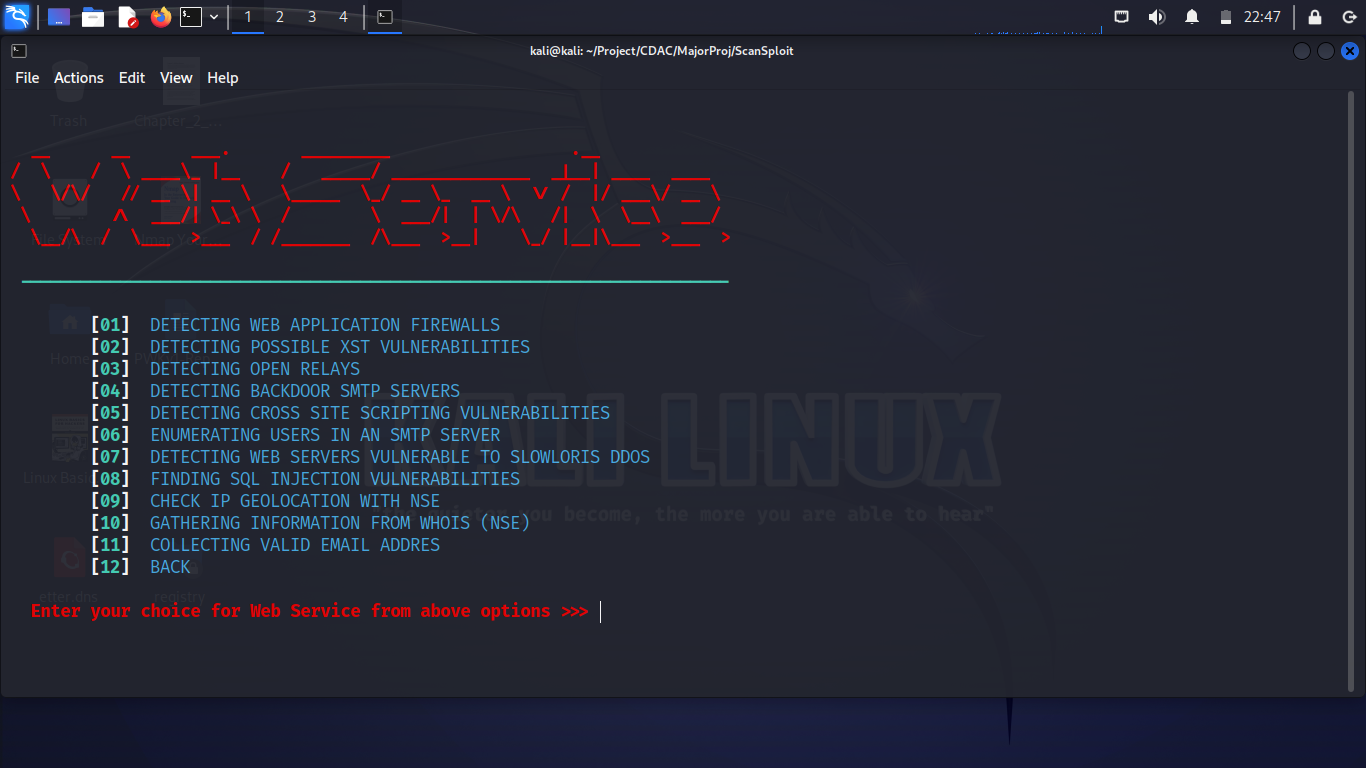
Nmap Figure 5.2



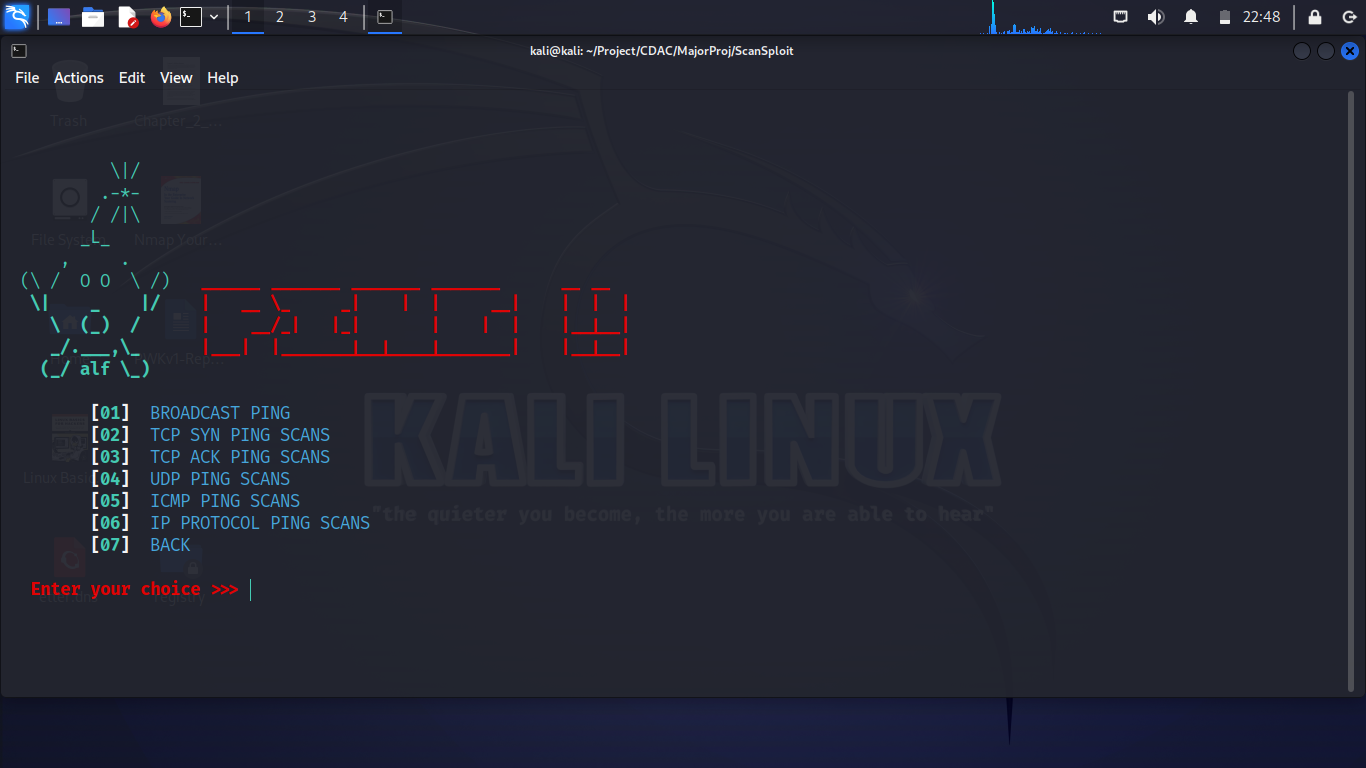
Nmap Figure 5.3



Nmap Figure 5.4



Nmap Figure 5.5



Nmap Figure 5.6

# References

6.1 <https://nmap.org>

6.2 <http://scanme.nmap.org>

6.3 <https://nmap.org/book/man-port-scanning-techniques.html>

6.4 <https://github.com/Screetsec/Dracnmap>

6.5 <https://www.digitalocean.com/community/tutorials/nmap-switches-scan-types#1-tcp-connect-scans-st>

6.6 <https://www.redhat.com/sysadmin/nmap-scripting-engine>

6.7 <https://nmap.org/book/nse.html>