# **Table of Contents**

1. Legal Notices
2. What is Nmap?
3. Recommended Knowledge
   1. CIDR notation
   2. TCP / UDP Networking
      1. TCP Operation
      2. UDP Operation
   3. Common Firewall Types
   4. Port States
4. Nmap Syntax Basics
5. Host Discovery
6. Protocol Based Options
   1. TCP Options
   2. UDP Options
7. Port Specification and Scan Order
8. Nmap Scripts and Service Scanning
9. OS Detection
10. Output Options
11. Firewall Evasion
    1. Timing
    2. Spoofing
12. Useful Pre-made commands

# 

# **Legal Notices**

# The *nmap reference guide* (C) 2005–2020 Insecure.Com LLC is licensed using the [Creative Commons Attribution License v3.0](https://creativecommons.org/licenses/by/3.0/) and was originally created by Gordon "Fyodor" Lyon in 1997 and has since then been developed by hundreds of people. David Fifield and Daniel Miller especially for their multi-year contributions. The original guide can be found [here.](https://nmap.org/book/man.html)

# Please note that in some states and countries performing scans on networks you do not own or have **EXPLICIT, WRITTEN** permission from is considered illegal. Just because in later sections this guide at length shows how to perform scans that are harder to detect, know that the scans are still visible and are still trackable.

### **Even in states or countries where it is legal, your ISP can and is sometimes compelled legally to ban you.**

### **Do *NOT* under any circumstances scan machines or networks that are not yours.**

# I take no responsibility for the actions of others who may have used this guide.

# Last Update:

This Document was last updated using the October 24th, 2020 wiki page.

For the most up-to-date documentation, either visit the google docs link provided in private, or this [GitHub wiki.](https://github.com/JGundy64/using-nmap/wiki)

# 

# What is Nmap

### **From the official nmap website:**

“Nmap (‘Network Mapper’) is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.”

#### [**Source**](https://nmap.org/book/man.html#man-description)

### **In other words**

Nmap is an application that can tell you loads about your network, a remote network, and what systems and services are running on either of them. In the scope of a security testing, this is great information for determining many things; what hosts are up, what services are exposed on a host, is it vulnerable, and much more.

# Recommended Knowledge

This is a section dedicated to explaining network fundamentals and acronyms used in the wiki. If you are comfortable with the topics listed below, feel free to move onto the next section of the wiki.

## **CIDR Notation**

CIDR stands for Classless Inter-Domain Routing and was introduced by the [IETF](https://www.ietf.org/) in 1993 ([Original RFC](https://tools.ietf.org/html/rfc1518)) to replace classful networking on the internet. All modern IP addresses consist of two things; the host address (the address of the computer or endpoint) and the network address (or subnet). These two items tell a router or switch where a packet of information needs to go on a network or two another network.

CIDR notation is designed to make host and network address writing easier to write down. For our purposes, a network address is what tells a router or computer what network a given host is on and the host IP is the address of the computer on the network you're attempting to communicate to.

The composition of a CIDR address is the network or host IP and the number of bits enabled in the subnet mask. IPv6 CIDR calculations will not be covered in this wiki at this time, but here is [a resource](https://www.calculator.net/ip-subnet-calculator.html) that can do so for you. In order to find the network address of a computer, you must figure out how many bits are turned on in the subnet mask, and then take those same leading bits (including those not enabled) in the host address and add them together in each octet provided.

Now, using a binary table for 8 bits we can find the number of bits turned on in each octet of the address. If starting from a network mask, since 255 is all bits enabled in IPv4 subnet masking for each 255 in your mask simply represent it with an 8 until you have a mask with a value less than 255, then put that value into the binary table. If starting from a CIDR bit notation, simply divide by 8 until a remainder is reached, that remainder represents the number of bits enabled in series from left to right in the tables below.

Two binary tables with examples have been included below for clarity.

#### 

#### **If given a raw subnet value:**

Given: 255.255.248.0

1. Since 255 will always equal 8 bits enabled, we know the subnet to be at least 8 + 8 + x + 0.
2. For the third octet, subtract the left most value in the table below until you have no remainder.
3. Every time you subtract, place a one in the "bits enabled" row.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Powers (base 2)** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Bits Enabled | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

1. Now count how many 1s are in the table.
2. Next replace x in the addition problem in step 1.
3. The subnet in CIDR is written with a "/" in front, so your subnet is /21!

#### 

#### **If given CIDR shorthand value:**

Given a subnet of /19:

1. Divide by 8.
2. Each division with no remainder is a full octet, we have two, so we know the first two octets of the subnet: 255.255.x.x.
3. For the remainder, place that many 1s from left to right in the bits enabled table below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Powers (base 2)** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Bits Enabled | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

1. Add the number to the next empty octet (from left to right) to get 255.255.224.x
2. Since no value after the first zero in a subnet can be enabled, do not add any more numbers.
3. Adding 128 + 64 + 32 + 16 gives us 224
4. Our /19 subnet mask must be: 255.255.224.0

### 

### **Now that we have figured out how to calculate subnets**

How do we use the subnet information to determine a network address? We do a similar process as mentioned above, but instead of comparing against a binary table, we compare against our subnet first. A step by step walkthrough is provided below:

#### **Example**

Given: Host = 10.10.216.120 and Subnet = /20

1. Find which octet the subnet mask ends in (divide the CIDR by 8, rounded up to next whole number, in this case 20 / 8 = 3)
2. Calculate the number of bits in the last octet of the subnet (As demonstrated above)
3. Calculate the number of bits enabled in the host address' corresponding octet (octet 3)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Powers (base 2)** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Bits Enabled in Host Address | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| Bits Enabled in Subnet Mask | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

1. To calculate the network address, find all cells where there is both a 1 in the host address AND a 1 in the subnet mask's enabled bits
2. Add the corresponding values.
3. Place this value in the octet that we are focused on
4. Ignore all octets after the octet we just performed our math on
5. Write the new IP address all together and add our original CIDR notation
6. In this example, our network address is 10.10.208.0/20

### **Common Subnets and their CIDR Values:**

|  |  |
| --- | --- |
| **Subnet Mask** | **CIDR Value** |
| 255.0.0.0 | /8 |
| 255.255.0.0 | /16 |
| 255.255.255.0 | /24 |

## **TCP / UDP Networking**

TCP and UDP are two ways of accomplishing the same goal: transmission of data.

The Main difference between TCP and UDP as network protocols is the fact that TCP guarantees data transmission through various means, whereas UDP simply throws the information towards a computer and hopes that it is received and processed.

### **TCP Operation**

One of the big advantages that TCP gives over UDP is the guarantee that all information is transmitted and is not altered in transmission. This is accomplished with checksums and sequence numbers embedded into every packet's header.

In order to start a TCP connection, both ends must perform what's commonly referred to as the TCP 3-way handshake using bits set in certain places, called flags.

This handshake from a client to a server consists of 3 packets containing certain flags set in the packet in this order:

1. SYN (to server)
2. SYN ACK (to client)
3. ACK (to server then alternating)

After this process has been performed, the computers simply continue to communicate with ACK messages containing various information inside of them and may set other special flags covered elsewhere in the wiki.

Ending a TCP connection is done through the use of the FIN flag and a very similar process as was just shown above:

1. FIN (to server)
2. ACK FIN (to client)
3. ACK (to server, connection is now closed)

*Closing* can be performed at any time by *either side,* but *opening must be done by the client*.

If a TCP connection drops a packet, a rebroadcast request is sent to the host which the packet should have originated from asking the host to send, starting from the dropped packet, in sequence again.

If a TCP connection closes suddenly and/or a server being connected to is talked with out of sequence, the server will respond with an RST (reset) flag set in a packet, telling the client to restart the conversation over again with the TCP 3-way handshake.

### **UDP Operation**

The advantage that UDP provides over TCP is speed. A UDP packet is much smaller and faster to process and transmit, but may not make it intact or in order if at all.

The only things carried by a UDP packet are the source port, destination port, data, and (optionally in IPv4) a checksum.

There is no connection established in UDP communication, data is simply sent from client to server and visa-versa on a certain port and is hopefully processed.

## **Common Firewall Types and Operation**

There are two types of firewalls, Stateful and Stateless.

Some key differences between them is speed, resource usage, and ability to fine-tune rules.

### **Stateful**

Stateful firewalls provide the following key things:

1. Constant analysis of all connections in a network
2. Contextual packet filtering based on connection status
3. Data contained within a connection

A stateful firewall keeps a log of all current connections on a network, what state those connections are in, and sometimes what kind of data is being transmitted. The biggest downside to stateful firewalls are that they're slower, are dramatically more resource intensive, and oftentimes have to be separate, dedicated systems to operate quickly without impacting users. Another upshot to stateful firewalls is that they detect when TCP packets are sent out of sequence, and can stop some specific kinds of attacks as mentioned elsewhere on the wiki.

Common examples of stateful firewalls are:

1. pfSense
2. Sophos XG
3. Windows Defender Firewall

### 

### **Stateless**

Stateless firewalls provide the following:

1. Speedy firewall rulings
2. Resource efficient firewall actions
3. Low impact to latency behind the firewall

The key benefits of using stateless firewalls are that they take very little resources to run, are comparatively very fast when enforcing their rules, and simple to make rules for. Even a system as small as a raspberry pi can handle running a stateless firewall without seeing a dramatic performance impact. However, they are sometimes vulnerable to some very specific TCP-based workarounds that may allow scans around a firewall.

Common examples of stateless firewalls are:

1. UFW / iptables
2. Cisco Meraki

## **Port States Supported by Nmap**

According to the nmap website, nmap supports 6 different port states.

The states nmap recognizes are:

1. Open
2. Closed
3. Filtered
4. Unfiltered
5. Open|Filtered
6. Closed|Filtered

They are described in great detail in the Official documentation and I would be basically copying them by writing it here, so please follow [this link](https://nmap.org/book/man-port-scanning-basics.html) if you have doubts of what those terms mean.

# Nmap Syntax Basics

The syntax for nmap is fairly straight forward:

nmap [ Scan Type ] [ Options ] { target specification }

##### [**Source**](https://nmap.org/book/man.html)

## **What is required?**

The only two items that are required above for a scan to commence are the name of the command and the target specification.

Simply supplying nmap with a target machine or network is enough to perform a basic scan on a host or network and will tell you if a host is up and what ports are open.

# Host Discovery

Nmap uses 3 main methods of discovering if a host is online. The default way is by sending out an ARP request to all addresses, in a random order, within a range supplied, or at a single host IP address. Another way is to simply use ICMP Echo requests (aka ping) against the target system or network and see who replies. Alternatively, you can tell nmap to treat a host as online and do your own port analysis by seeing what ports are open at a given IP.

### **ARP Scanning (-sL or no options)**

In order to use an ARP request scan, use the -sL option or no option as -sL is the default scan option. This scan, after finding a host that is online, will also scan the top 1000 most commonly used ports on a host to see if any respond to TCP or UDP packets on those ports. Some modern IDS/IPS systems will create an alert for this kind of activity, but it is not instantly suspect activity due to all routers and some switches using ARP requests to build internal network maps.

### **Ping Scanning (-sn)**

In order to ping all hosts and not use an ARP request, use to -sn option. This option *does not* scan an online host's ports and can be blocked by certain firewall rules. This is a very fast scan as it does not perform port scanning afterwards, but may not see every single system. Many IDS/IPS systems will detect this instantly if not whitelisted and will raise an alarm instantly if not outright banning the scanning system. However, some networks have heartbeat detection on hosts in the form of pings, so ping scanning may actually be more stealthy than ARP scanning.

### **Port Analysis *Only* (-Pn)**

In order to skip the host discovery step completely and assume all hosts supplied are online, you must use the -Pn flag. This flag is useful if the previous two methods are blocked or getting your system banned, but you know a host is online at a certain address and has at least some service running.

[**Source**](https://nmap.org/book/man-host-discovery.html)

# Protocol Based Options

Through nmap, you can use certain options to manipulate the flags set in a TCP packet or sending only UDP datagrams. These various options can be used to receive different responses depending on the system, service, or firewall type in place as explained below.

## **TCP Packet Options**

### **NULL Packets (-sN)**

A null packet is simply a TCP packet that has no flags set at all. Different combinations of services, firewalls, and sometimes operating systems produce different responses to these packets. This flag is useful for sometimes bypassing stateless firewalls that may only check if a packet *doesn't have* a specific flag or set of flags set. As alluded to above, sometimes systems will mistakenly send response packets even if they shouldn't respond to any requests at all.

### **SYN (-sS)**

A SYN packet is a TCP packet with *only* the SYN flag set. This simulates the start of a TCP conversation, and will almost always start a TCP conversation with a system that responds to TCP connections. However, some stateless firewalls will filter packets like this originating from outside of their local network. This is the default port scanning option.

### **Connect (-sT)**

This is very similar to a SYN scan, but instead of nmap directly manipulating the packet's SYN flag on, it requests the scanning computer's OS to start a connection with the remote host on a specific port. This is the alternate default if SYN scanning is not available.

### **ACK (-sA)**

ACK scans attempt to jump right into the middle of the TCP connection and cause the remote host to send an RST packet. This kind of scan is rejected or dropped by nearly all properly configured stateful firewalls and can lead to false results if a stateful firewall is encountered. However, most if not all stateless firewalls will not block these regardless of connection status and the service will send a TCP RST packet.

### **FIN (-sF)**

FIN scans have a similar effect to ACK scans, but instead use the end of a conversation instead of the middle. This is likely dropped by most if not all stateful firewalls if no connection state proper to receiving a FIN packet is currently being used by the scanning system.

### **XMAS (-sX)**

The XMAS Scan is a combination of strange flags set in a TCP packet, the FIN, PSH, and URG flags. These flags are the flags used to tell routers that this message is incredibly urgent and time sensitive while closing a non-existent conversation. Again this is a scan likely to be dropped by most stateful firewalls but also is able to be detected by stateless firewalls if configured to do so as this set of flags is incredibly rare amongst legitimate traffic.

### **Sliding Window Exploit (-sW)**

This method of port scanning can sometimes be used to make the distinction between closed and filtered ports on some firewalls. This method of scanning relies on the fact that some firewalls will return a sliding window value of 1 on filtered ports, but a 0 on closed ports. This can be useful, sometimes, for detecting if your system is simply not able to connect to a port, but some other system might be. This scan uses

## **UDP Packet Options**

### **UDP Only (-sU)**

This option causes nmap to only check ports using UDP datagrams. Some services, such as DNS queries, only need UDP to operate and indicate a running service on a target system or network. This flag can help evade some certain stateful firewalls as there is no connection established in communication when trying to check for some services.

#### [**Source**](https://nmap.org/book/man-port-scanning-techniques.html)

# Port Specification and Scan Order

Nmap allows you to not only specify what ports to scan, but also the order that they are scanned in.

By default, nmap will scan the top 1000 most commonly used ports and in a random order. This is oftentimes the best balance between complete scanning and time investment, as well as the randomization helping to avoid IDS/IPS systems by default.

## **Port Range (-P)**

You can specify a port range by hand using the -P option. This is useful for only scanning for a limited set of services that are known to operate on a specific few ports. Using this to narrow down how many ports are scanned can drastically speed up the scan process.

## **Fewer Ports (-F)**

The -F option acts similar to the -P option, but will only scan [IANA assigned ports.](https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml)

## **Top ports (--top-ports [ number ])**

This option is a tunable version of nmap's default configuration and is a great way to speed up scans without needing to know in advance exactly what range of ports a system's services are running in.

## **Do *Not* Randomize (-r)**

This option stops nmap from randomizing the order in which port scans are performed. Do note that even *without* this option enabled, all results will be in order from lowest to highest.

#### [**Source**](https://nmap.org/book/man-port-specification.html)

# Nmap Scripts and Service Scanning

Nmap comes with a powerful scripting engine built into it which has some very useful scripts installed by default.

## **Service Scanning (-sV)**

The service scanning script probes version and service information on the remote system. This script does not enumerate vulnerabilities on a system, but enhances what information will be gathered by the scanner. This is oftentimes noticed by IDS/IPS systems and will generate an alert, but not always a ban.

# **Scripts**

## **Default (-sC)**

The default set of scripts are described as "generally useful scripts" that are supposed to be non-intrusive, speedy, reliable and not incredibly verbose. This set of scripts is often a starting point for figuring out what services are on a host or a network, and is incredibly useful with very little user interaction or tuning needed.

## **Vulnerabilities (--script vuln)**

The vulnerability detection scripts are a set of scripts that attempt to find and enumerate vulnerabilities on a system or network. These scripts are almost always picked up as a scan by IDS systems and sometimes will trigger an automated ban, but are incredibly useful in determining if a host is vulnerable and what exactly it is vulnerable to. This script can be almost too verbose at times, and comes with a lot of information and thus takes a comparatively long time to complete.

## **Safe (--script safe)**

The safe set of scripts are designed to augment a scan without being intrusive to the remote host. These scripts are mostly general network discovery similar to a system being connected to a network for the first time. These scripts also are designed to try to not crash or exploit anything on a remote host and are sometimes a little harder for IDS systems to detect, especially when used against a single host.

## **Discovery (-script discovery)**

The discovery scripts are designed to augment the network discovery nmap can provide by querying public data in a network such as: SMB shares, HTML titles, SNMP system details, etc. This is also sometimes not picked up by IDS systems, and almost never will lead to a ban simply because all of this information should be public on a network.

## **Broadcast (-script broadcast)**

This script helps nmap find hosts that will only respond to broadcast messages. I've found this acts similar to the default scan method using ARP requests.

## **All Except \_\_\_ (-script "not \_\_\_")**

This syntax combination will load all scripts except a specific script or list of scripts.

## **Script A and Script B (-script "A and B" [or] -script A,B)**

This syntax combination will load multiple scripts and only the scripts listed.

There are many more scripts and use cases, but some are incredibly specific and not as commonly useful as the ones listed above. However, you can find all scripts at the source.

### [**Source**](https://nmap.org/book/nse-usage.html#nse-categories)

# OS Detection

Nmap has the ability to perform OS detection through TCP/IP fingerprinting. This is an amazing, though not 100% accurate, feature and is very helpful for figuring out what operating system the remote host is running.

The option to invoke nmap's OS detection is "-O".

#### [**Source**](https://nmap.org/book/man-os-detection.html)

# Nmap Output Options

Nmap only has 3 (serious) output methods. I strongly recommend that, when performing a scan, you *always* output your results to a file just in case something strange happens.

## **Normal (-oN [ filename ])**

The "Normal" output is the same exact output and formatting that is displayed in the terminal window that nmap was run inside of. This option outputs a raw text file.

## **XML (-oX [ filename ])**

The XML output format creates an xml file that holds the same information as a normal file, just formatted as an XML sheet to be viewed in a compatible program at a later date.

## **Grepable (-oG [ filename ])**

The grepable output format outputs the same information as normal, is still as raw text, but is able to be more easily searched using the grep command on Linux. This can be helpful for command line environments or those of us who enjoy using grep to sort through files quickly. It also helps when trying to create a script to read your outputted file and tell you relevant information quickly.

#### [**Source**](https://nmap.org/book/man-output.html)

# Firewall Evasion

Oftentimes, the methods that nmap uses to scan are blocked by a firewall in various places. This can be by a router's built-in firewall, the remote host's firewall, a dedicated firewall, and anything in between. For this reason, it may be necessary to perform some more invasive methods to evasion than discussed previously to get an accurate scan result.

## **Timing (-T [ 0-5 ] or --max-rate [ number ])**

One general thing that most firewall and IDPS systems look for when determining what is and is not good traffic is the rate at which data is coming in. Sometimes, nmap's default setting is fast enough to trigger small rate-limiting bans on your scanning system. The "-T" Option is used to tell nmap a relative speed to send packets at, by default 3. The "--max-rate" option is used to set the exact rate at which nmap will send packets per second to the remote host. Although slowing down the rate at which the scan completes, this can help us avoid banning our network scanner.

## **Packet Fragmentation (-f)**

Packet fragmentation refers to, in this case, splitting up the nmap's packet header amongst multiple packets instead of one. Since stateful firewalls do sometimes inspect packet headers, this can help in avoiding some firewall bans.

## **IP Decoys or Spoofs (-D [ IP1 ], [ IP2 ], [ ME ], etc... )**

Using the decoy option creates fake scans originating from either fake or legitimate IP addresses against a network. This can be helpful for network stress testing or causing some firewall systems to not ban your real IP for a little while. Using "ME" allows you to set the position in the scan in which your real IP will be used. Doing so at later numbers makes it less likely to be detected or treated as the real scanner.

**Sources**:

[Timing](https://nmap.org/book/man-performance.html)

[Fragmentation and Decoys](https://nmap.org/book/man-bypass-firewalls-ids.html)

# Pre Made Commands

This is by no means an exhaustive list of every single potentially useful command and option combination nmap has, but here are a few generic scans that I have found incredibly useful during Cyber Defense Competition specifically. Remember to use CIDR notation when inputting network addresses!

## **nmap -sL -oN hosts [ network address ]**

This is a command that is useful for simply figuring out what systems are on a network. This scan does not send any packets to the targeted hosts, but instead uses ARP requests to figure out what systems are online because they will tell us who they are. All output in the terminal will be copied to the file "hosts" in the directory in which you ran nmap inside of, the file will be created if it does not exist, and overwritten if it does.

## **nmap -sn -oN hosts [ network address ]**

This command is useful for quick and dirty scans that you know won't be interrupted by an IDPS system or firewall. it will simply ping all hosts in the provided network. All output in the terminal will be copied to the file "hosts" in the directory in which you ran nmap inside of, the file will be created if it does not exist, and overwritten if it does.

## **nmap --top-ports 100 -sV -O -oN services [ host ]**

This command is a faster way to scan a host's services and gain as much information on a specific host within a relatively short amount of time. This set of options will only scan the top 100 most commonly used ports, figure out as much service information as it can from them, and attempt to figure out the host's operating system information. All output in the terminal will be copied to the file "services" in the directory in which you ran nmap inside of, the file will be created if it does not exist, and overwritten if it does.

## **nmap -Pn --script vuln -O -oN vulns [ host ]**

This set of options is used in vulnerability enumeration. This scan assumes the remote host is online, will list out all vulnerabilities and services found on a host, the OS information for easier identification later, and copies all terminal output to a text file called "vulns" in the directory from which you ran nmap from.