Enumeration with *nmap*

Enumeration is the most critical part of all. The art, the difficulty, and the goal are not to gain access to our target computer. Instead, it is identifying all of the ways we could attack a target we must find.

Most of the ways we can get access to a system can be narrowed down to the following two points:

- Functions and/or resources that allow us to interact with the target and/or provide additional information.
- Information that provides us with even more important information to access our target.

When scanning and inspecting, we look exactly for these two possibilities. Most of the information we get comes from misconfigurations or neglect of security for the respective services. Misconfigurations are either the result of ignorance or a wrong security mindset. For example, if the administrator only relies on the firewall, Group Policy Objects (GPOs), and continuous updates, it is often not enough to secure the network.

Enumeration is the key.

Introduction to *nmap*

Network Mapper (Nmap) is an open-source network analysis and security auditing tool written in C, C++, Python, and Lua. It is designed to scan networks and identify which hosts are available on the network using raw packets, and services and applications, including the name and version, where possible. It can also identify the operating systems and versions of these hosts. Besides other features, Nmap also offers scanning capabilities that can determine if packet filters, firewalls, or intrusion detection systems (IDS) are configured as needed.

Use Cases

The tool is one of the most used tools by network administrators and IT security specialists. It is used to:

- Audit the security aspects of networks
- Simulate penetration tests
- Check firewall and Intrusion Detection System (IDS) settings and configurations
- Network mapping
- Response analysis
- Identify open ports
- Vulnerability assessment

Nmap Scanning Modes

Nmap offers many different types of scans that can be used to obtain various results about our targets. Basically, Nmap can be divided into the following scanning techniques:

- Host discovery
- Port scanning
- Service enumeration and detection
- OS detection
- Scriptable interaction with the target service (Nmap Scripting Engine)

Syntax

The syntax for Nmap is fairly simple and looks like this: `nmap <scan types> <options> <target>`

SCAN TECHNIQUES:

- -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
- -sU: UDP Scan
- -sN/sF/sX: TCP Null, FIN, and Xmas scans
- –scanflags : Customize TCP scan flags
- -sI <zombie host[:probeport]>: Idle scan
- -sY/sZ: SCTP INIT/COOKIE-ECHO scans
- -sO: IP protocol scan
- -b : FTP bounce scan

For example, the TCP-SYN scan (-sS) is one of the default settings unless we have defined otherwise and is also one of the most popular scan methods. This scan method makes it possible to scan several thousand ports per second. The TCP-SYN scan sends one packet with the SYN flag and, therefore, never completes the three-way handshake, which results in not establishing a full TCP connection to the scanned port.

- If our target sends an SYN-ACK flagged packet back to the scanned port, Nmap detects that the port is open.
- – If the packet receives an RST flag, it is an indicator that the port is closed. If Nmap does not receive a packet back, it will display it as filtered. Depending on the firewall configuration, certain packets may be dropped or ignored by the firewall. Let us take an example of such a scan.

Exercise

- Run an open port scan on the localhost
- `sudo nmap -sS localhost`

Use Cases in Depth

Open Port Scanning

```
nmap -sS <host>
```

Host Discovery

When we need to conduct an internal penetration test for the entire network of a company, for example, then we should, first of all, get an overview of which systems are online that we can work with. To actively discover such systems on the network, we can use various Nmap host discovery options. There are many options Nmap provides to determine whether our target is alive or not. The most effective host discovery method is to use *ICMP echo requests*, which we will look into.

It is always recommended to store every single scan. This can later be used for comparison, documentation, and reporting. After all, different tools may produce different results. Therefore it can be beneficial to distinguish which tool produces which results.

```
sudo nmap 10.129.2.0/24 -sn -oA tnet | grep for | cut -d" " -f5

10.129.2.4

10.129.2.10

10.129.2.11

10.129.2.18

10.129.2.19

10.129.2.20

10.129.2.28
```

Flag	Description
-sn	disables port scanning
-oA tnet	stores results in various formats, files named tnet

Exercise Try to familiarize yourself with the command. What does it do and why?

We can also scan for multiple IPs: - name them one after another, e.g. in `sudo nmap -sn 10.129.2.18 10.129.2.19` - use an input file, e.g. hosts.lst listing IPs separated with a line feed via `sudo nmap -sn -iL hosts.lst`

If we disable port scan (-sn), Nmap automatically ping scan with ICMP Echo Requests (-PE). Once such a request is sent, we usually expect an ICMP reply if the pinging host is alive. The more interesting fact is that our previous scans did

not do that because before Nmap could send an ICMP echo request, it would send an ARP ping resulting in an ARP reply. We can confirm this with the "–packet-trace" option. To ensure that ICMP echo requests are sent, we also define the option (-PE) for this.

```
sudo nmap 10.129.2.18 -sn -oA host -PE --packet-trace

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 00:08 CEST

SENT (0.0074s) ARP who-has 10.129.2.18 tell 10.10.14.2

RCVD (0.0309s) ARP reply 10.129.2.18 is-at DE:AD:00:00:BE:EF

Nmap scan report for 10.129.2.18

Host is up (0.023s latency).

MAC Address: DE:AD:00:00:BE:EF

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
```

Flag	Description
-PE	sends ICMP Echo requests
–packet-trace	shows all packets sent and received
–reason	provides reason for nmap's judgment

So, the judgment alive or dead is achieved by ARP pings. To enforce ICMP requests, we can use the `--diable-arp-ping`command.

```
sudo nmap 10.129.2.18 -sn -oA host -PE --packet-trace --disable-arp-ping
```

Host and Port Scanning

After we have found out that our target is alive, we want to get a more accurate picture of the system. The information we need includes:

- Open ports and its services
- Service versions
- Information that the services provided
- Operating system

Scanning Open Ports States for scanned ports

- *open*: This indicates that the connection to the scanned port has been established. These connections can be TCP connections, UDP datagrams as well as SCTP associations.
- *closed*: When the port is shown as closed, the TCP protocol indicates that the packet we received back contains an RST flag. This scanning method can also be used to determine if our target is alive or not.

- *filtered*:Nmap cannot correctly identify whether the scanned port is open or closed because either no response is returned from the target for the port or we get an error code from the target.
- *unfiltered*: This state of a port only occurs during the TCP-ACK scan and means that the port is accessible, but it cannot be determined whether it is open or closed.
- *open/filtered* If we do not get a response for a specific port, Nmap will set it to that state. This indicates that a firewall or packet filter may protect the port.
- *closed/filtered*: This state only occurs in the IP ID idle scans and indicates that it was impossible to determine if the scanned port is closed or filtered by a firewall.

By default, Nmap scans the top 1000 TCP ports with the SYN scan (`-sS`). This SYN scan is set only to default when we run it as root because of the socket permissions required to create raw TCP packets. Otherwise, the TCP scan (`-sT`) is performed by default. This means that if we do not define ports and scanning methods, these parameters are set automatically. We can define the ports one by one (`-p 22,25,80,139,445`), by range (`-p 22-445`), by top ports (`--top-ports=10`) from the Nmap database that have been signed as most frequent, by scanning all ports (`-p-`) but also by defining a fast port scan, which contains top 100 ports (`-F`). To have a clear view of the SYN scan, we disable the ICMP echo requests (`-Pn`), DNS resolution (`-n`), and ARP ping scan (`--disable-arp-ping`).

A typical answer could contain the following items:

- SENT (0.0429s): Indicates the SENT operation of Nmap, which sends a packet to the target
- RCVD (0.0573s): Indicates a received packet from the target
- TCP: Shows the protocol that is being used to interact with the target port
- 10.10.14.2:63090 >: Represents our IPv4 address and the source port, which will be used by Nmap to send the packets
- 10.129.2.28:21: Shows the target IPv4 address and the target port
- S: SYN flag of the sent TCP packet
- RA: RST and ACK flags of the sent TCP packet.
- ttl=56 id=57322 iplen=44 seq=1699105818 win=1024 mss 1460: Additional TCP Header parameters

Connect Scan The Nmap TCP Connect Scan (`-sT`) uses the TCP three-way handshake to determine if a specific port on a target host is open or closed. The scan sends an SYN packet to the target port and waits for a response. It is considered open if the target port responds with an SYN-ACK packet and closed if it responds with an RST packet.

The Connect scan is useful because it is the most accurate way to determine the

state of a port, and it is also the most stealthy. Unlike other types of scans, such as the SYN scan, the Connect scan does not leave any unfinished connections or unsent packets on the target host, which makes it less likely to be detected by intrusion detection systems (IDS) or intrusion prevention systems (IPS). It is useful when we want to map the network and don't want to disturb the services running behind it, thus causing a minimal impact and sometimes considered a more polite scan method.

It is also useful when the target host has a personal firewall that drops incoming packets but allows outgoing packets. In this case, a Connect scan can bypass the firewall and accurately determine the state of the target ports. However, it is important to note that the Connect scan is slower than other types of scans because it requires the scanner to wait for a response from the target after each packet it sends, which could take some time if the target is busy or unresponsive.

Example: `sudo nmap 10.129.2.28 -p 443 --packet-trace --disable-arp-ping -Pn -n --reason -sT`

Dealing with Filtered Ports When a port is shown as filtered, it can have several reasons. In most cases, firewalls have certain rules set to handle specific connections. The packets can either be dropped, or rejected. When a packet gets dropped, Nmap receives no response from our target, and by default, the retry rate (`--max-retries`) is set to 1. This means Nmap will resend the request to the target port to determine if the previous packet was not accidentally mishandled.

Let us look at an example where the firewall drops the TCP packets we send for the port scan. Therefore we scan the TCP port 139, which was already shown as filtered. To be able to track how our sent packets are handled, we deactivate the ICMP echo requests (`-Pn`), DNS resolution (`-n`), and ARP ping scan (`--disable-arp-ping`) again.

```
sudo nmap 10.129.2.28 -p 139 --packet-trace -n --disable-arp-ping -Pn
```

Answer:

```
SENT (0.0381s) TCP 10.10.14.2:60277 > 10.129.2.28:139 S ttl=47 id=14523 iplen=44  seq=4175236769 win=1024 <mss 1460>

SENT (1.0411s) TCP 10.10.14.2:60278 > 10.129.2.28:139 S ttl=45 id=7372 iplen=44  seq=4175171232 win=1024 <mss 1460>
```

We see in the last scan that Nmap sent two TCP packets with the SYN flag. The case is different if the firewall rejects the packets. For this, we look at TCP port 445, which is handled accordingly by such a rule of the firewall.

Answer:

```
SENT (0.0388s) TCP 10.129.2.28:52472 > 10.129.2.28:445 S ttl=49 id=21763 iplen=44  seq=1418633433 win=1024 <mss 1460>

RCVD (0.0487s) ICMP [10.129.2.28 > 10.129.2.28 Port 445 unreachable (type=3/code=3) ] IP [ttl=64 id=20998 iplen=72 ]
```

As a response, we receive an ICMP reply with type 3 and error code 3, which indicates that the desired port is unreachable. Nevertheless, if we know that the host is alive, we can strongly assume that the firewall on this port is rejecting the packets, and we will have to take a closer look at this port later.

Discovering Open UDP Ports Some system administrators sometimes forget to filter the UDP ports in addition to the TCP ones. Since UDP is a stateless protocol and does not require a three-way handshake like TCP. We do not receive any acknowledgment. Consequently, the timeout is much longer, making the whole UDP scan (`-sU`) much slower than the TCP scan (`-sS`).

Another disadvantage of this is that we often do not get a response back because Nmap sends empty datagrams to the scanned UDP ports, and we do not receive any response. So we cannot determine if the UDP packet has arrived at all or not. If the UDP port is open, we only get a response if the application is configured to do so.

Example: `sudo nmap 10.129.2.28 -sU -Pn -n --disable-arp-ping --packet-trace -p 137 --reason`

If we get an ICMP response with error code 3 (port unreachable), we know that the port is indeed closed. For all other ICMP responses, the scanned ports are marked as (open|filtered).

Another handy method for scanning ports is the -sV option which is used to get additional available information from the open ports. This method can identify versions, service names, and details about our target.

Service Enumeration

For us, it is essential to determine the application and its version as accurately as possible. We can use this information to scan for known vulnerabilities and analyze the source code for that version if we find it. An exact version number allows us to search for a more precise exploit that fits the service and the operating system of our target.

Service Version Detection It is recommended to perform a quick port scan first, which gives us a small overview of the available ports. This causes significantly less traffic, which is advantageous for us because otherwise we can be discovered and blocked by the security mechanisms. We can deal with these first and run a port scan in the background, which shows all open ports (`-p-`). We can use the version scan to scan the specific ports for services and their versions (`-sV`).

A full port scan takes quite a long time. To view the scan status, we can press the [Space Bar] during the scan, which will cause Nmap to show us the scan status.

Another option (`--stats-every=5s`) that we can use is defining how periods of time the status should be shown. Here we can specify the number of seconds (s) or minutes (m), after which we want to get the status.

We can also increase the verbosity level (`-v / -vv`), which will show us the open ports directly when Nmap detects them.

Once the scan is complete, we will see all TCP ports with the corresponding service and their versions that are active on the system.

Primarily, Nmap looks at the banners of the scanned ports and prints them out. If it cannot identify versions through the banners, Nmap attempts to identify them through a signature-based matching system, but this significantly increases the scan's duration. One disadvantage to Nmap's presented results is that the automatic scan can miss some information because sometimes Nmap does not know how to handle it. Let us look at an example of this.

Example: `sudo nmap 10.129.2.28 -p- -sV -Pn -n --disable-arp-ping --packet-trace`

Answer:

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 20:10 CEST

NSOCK INFO [0.4200s] nsock_trace_handler_callback(): Callback: READ SUCCESS for EID 18 [10.129.2.28:25] (35 bytes): 220 inlane ESMTP Postfix (Ubuntu)..

Service scan match (Probe NULL matched with NULL line 3104): 10.129.2.28:25 is smtp.  Version: |Postfix smtpd|||

NSOCK INFO [0.4200s] nsock_iod_delete(): nsock_iod_delete (IOD #1)

Nmap scan report for 10.129.2.28?

Host is up (0.076s latency)

PORT   STATE SERVICE VERSION

25/tcp open  smtp    Postfix smtpd

MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Service Info: Host:  inlane

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
```

If we look at the results from Nmap, we can see the port's status, service name, and hostname. Nevertheless, let us look at this line here:

- NSOCK INFO [0.4200s] nsock_trace_handler_callback(): Callback: READ SUCCESS for EID 18 [10.129.2.28:25] (35 bytes): 220 inlane ESMTP Postfix (Ubuntu)..

Then we see that the SMTP server on our target gave us more information than Nmap showed us. Because here, we see that it is the Linux distribution Ubuntu. It happens because, after a successful three-way handshake, the server often sends a banner for identification. This serves to let the client know which service it is working with. At the network level, this happens with a PSH flag in the TCP header. However, it can happen that some services do not immediately provide such information. It is also possible to remove or manipulate the banners from the respective services. If we manually connect to the SMTP server using nc, grab the banner, and intercept the network traffic using tcpdump, we can see what Nmap did not show us.

TCPDump: `sudo tcpdump -i eth0 host 10.10.14.2 and 10.129.2.28`

Netcat: `nc -nv 10.129.2.28 25`

You see then the thee-way handshake followed by PSH and ACK flags, where PSH states that the target server is sending data to us and with ACK simultaneously informs us that all required data has been sent.

Nmap Scripting Engine

Nmap Scripting Engine (NSE) is another handy feature of Nmap. It provides us with the possibility to create scripts in Lua for interaction with certain services. There are a total of 14 categories into which these scripts can be divided:

Category	Description
auth	Determination of authentication credentials
broadcast	Scripts, which are used for host discovery by broadcasting and the discovered hosts, can be automatically added to the remaining scans
brute	Executes scripts that try to log in to the respective service by brute-forcing with credentials
default	Default scripts executed by using the -sC option
discovery	Evaluation of accessible services
dos	These scripts are used to check services for denial of service vulnerabilities and are used less as it harms the services
exploit	This category of scripts tries to exploit known vulnerabilities for the scanned port
external	Scripts that use external services for further processing

Category	Description
fuzzer	This uses scripts to identify vulnerabilities and unexpected packet handling by sending different fields, which can take much time
intrusive	Intrusive scripts that could negatively affect the target system
malware	Checks if some malware infects the target system
safe	Defensive scripts that do not perform intrusive and destructive access
version	Extension for service detection
vuln	Identification of specific vulnerabilities

Example with specific scripts: `sudo nmap 10.129.2.28 -p 25 --script banner,smtp-commands`

Aggressive scan: `sudo nmap 10.129.2.28 -p 80 -A`

Vulnerability Scan: `sudo nmap 10.129.2.28 -p 80 -sV --script vuln`