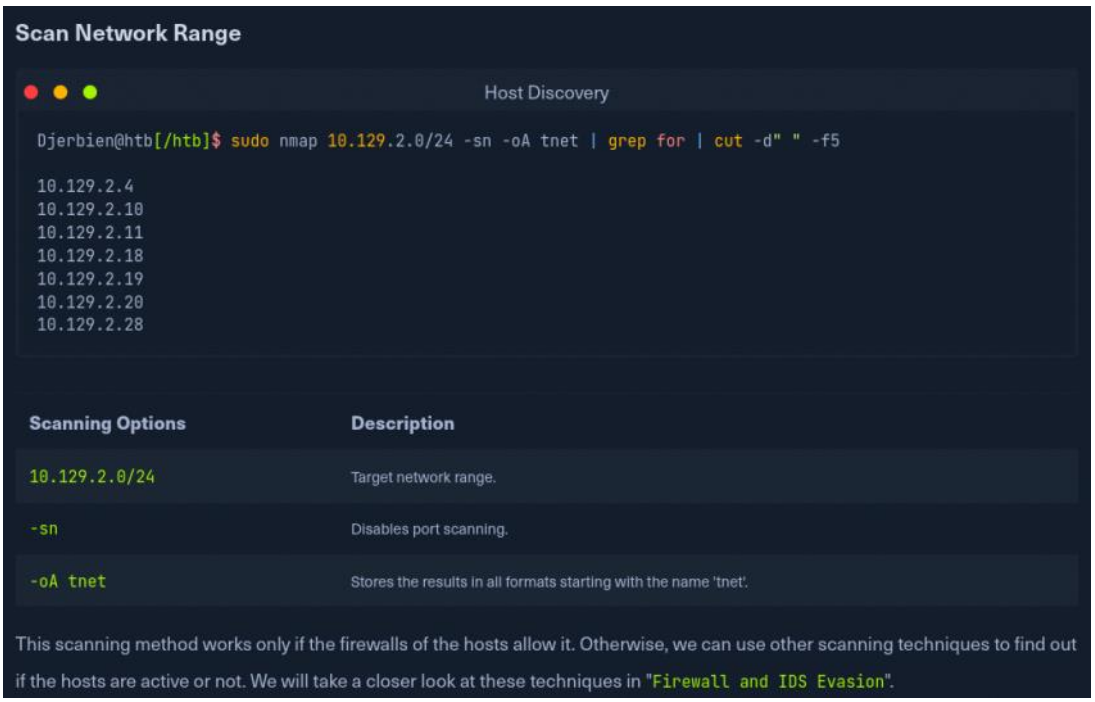
Network Enumeration with Nmap

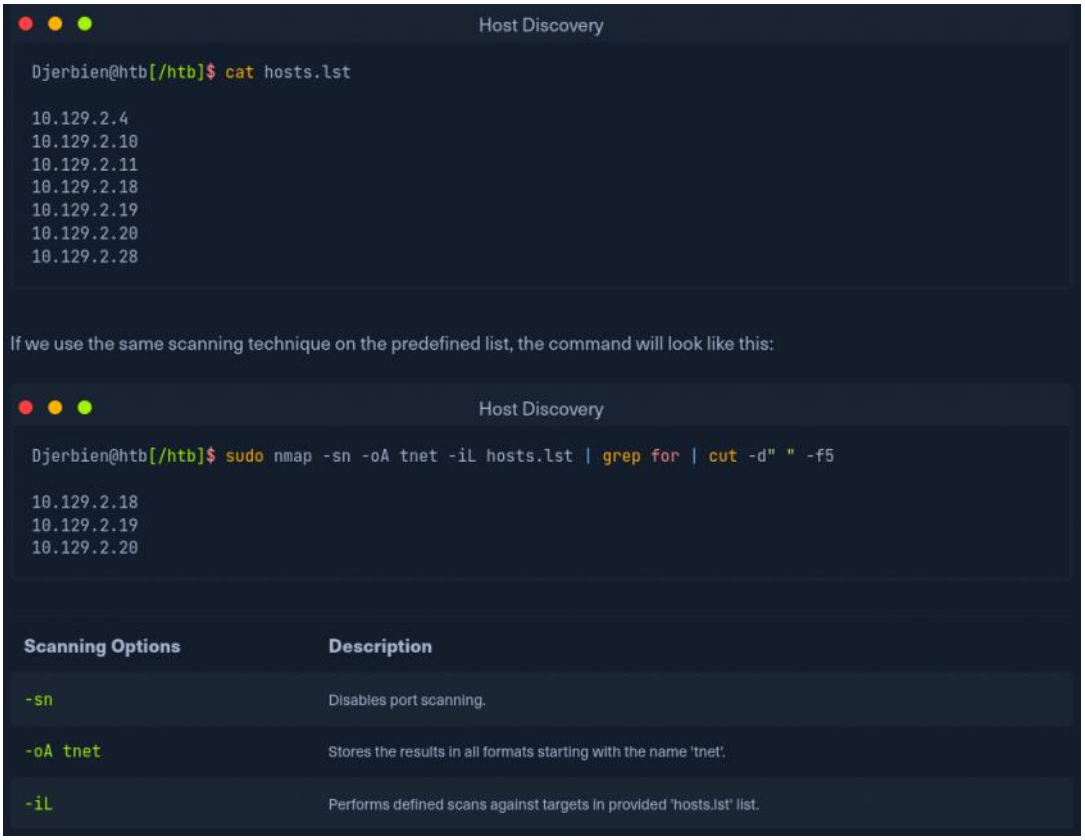
mercredi 11 septembre 2024 11:16 AM

Host Discovery

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



If we are given a list of ip adresses in the begining of the assessment, we may put them in txt file and then scan for the up hosts :



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.

Host Discovery

```
Djerbien@htb[/htb]$ 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
```

Scanning Options	Description
10.129.2.18	Performs defined scans against the target.
-sn	Disables port scanning.
-oA host	Stores the results in all formats starting with the name 'host'.
-PE	Performs the ping scan by using 'ICMP Echo requests' against the target.
--packet-trace	Shows all packets sent and received

Another way to determine why Nmap has our target marked as "alive" is with the "`--reason`" option.

Host Discovery

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.18 -sn -oA host -PE --reason

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 00:10 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, received arp-response (0.028s latency).
MAC Address: DE:AD:00:00:BE:EF
Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

Scanning Options	Description
10.129.2.18	Performs defined scans against the target.
-sn	Disables port scanning.
-oA host	Stores the results in all formats starting with the name 'host'.
-PE	Performs the ping scan by using 'ICMP Echo requests' against the target.
--reason	Displays the reason for specific result.

To disable the scan with ARP and use only ICMP :

We see here that Nmap does indeed detect whether the host is alive or not through the ARP request and ARP reply alone. To disable ARP requests and scan our target with the desired ICMP echo requests, we can disable ARP pings by setting the "--disable-arp-ping" option. Then we can scan our target again and look at the packets sent and received.

Host Discovery

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.18 -sn -oA host -PE --packet-trace --disable-arp-ping

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 00:12 CEST
SENT (0.0107s) ICMP [10.10.14.2 > 10.129.2.18 Echo request (type=8/code=0) id=13607 seq=0] IP [ttl=255 id=2
RCVD (0.0152s) ICMP [10.129.2.18 > 10.10.14.2 Echo reply (type=0/code=0) id=13607 seq=0] IP [ttl=128 id=406
Nmap scan report for 10.129.2.18
Host is up (0.086s latency).
MAC Address: DE:AD:00:00:BE:EF
Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

From the icmp reply we can determine the os :

TTL Values

The TTL value varies depends on the version of an operating system and device.

> The default initial TTL value for **Linux/Unix** is **64**, and TTL value for **Windows** is **128**.

Here is the default initial TTL values for popular operating systems such as Linux, FreeBSD, Mac OS, Solaris and Windows.

Host and Port Scanning

There are a total of 6 different states for a scanned port we can obtain:

State	Description
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.

Discovering Open TCP Ports

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).

Default Scans and Privileges

1. **SYN Scan (`-sS`)**:
   - **Default with Root Privileges**: When you run Nmap as root (using `sudo` or as the root user), it uses the SYN scan (`-sS`) by default.
   - **How it Works**: This scan method is also known as a "half-open" scan. Nmap sends SYN packets to ports and listens for responses. If a port responds with a SYN-ACK, it means the port is open. If it responds with RST, it means the port is closed. This scan is stealthy because it doesn't complete the TCP handshake.
   - **Why Root is Needed**: SYN scans require raw socket access to send and receive packets, which is a privileged operation in most operating systems.
2. **TCP Connect Scan (`-sT`)**:
   - **Default Without Root Privileges**: When you run Nmap as a non-root user, it defaults to the TCP connect scan (`-sT`).
   - **How it Works**: This method completes the full TCP handshake (SYN, SYN-ACK, ACK). It establishes a connection to the port and then closes it. This is less stealthy and more detectable but doesn't require raw socket access.
   - **Why It's Used**: Non-root users don't have the permissions needed for raw sockets, so Nmap falls back to using the standard system calls for establishing TCP connections.

Summary

- **With Root Privileges**: Nmap performs a SYN scan by default. It's faster and stealthier because it doesn't complete the full TCP handshake.
- **Without Root Privileges**: Nmap performs a TCP connect scan by default, as it can't use raw sockets. This scan is more easily detected because it completes the TCP handshake.

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -p 21 --packet-trace -Pn -n --disable-arp-ping

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 15:39 CEST
SENT (0.0429s) TCP 10.10.14.2:63090 > 10.129.2.28:21 S ttl=56 id=57322 iplen=44  seq=1699105818 win=1024 <m
RCVD (0.0573s) TCP 10.129.2.28:21 > 10.10.14.2:63090 RA ttl=64 id=0 iplen=40  seq=0 win=0
Nmap scan report for 10.11.1.28
Host is up (0.014s latency).

PORT   STATE  SERVICE
21/tcp closed ftp
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

Scanning Options	Description
`10.129.2.28`	Scans the specified target.
`-p 21`	Scans only the specified port.
`--packet-trace`	Shows all packets sent and received.
`-n`	Disables DNS resolution.
`--disable-arp-ping`	Disables ARP ping.

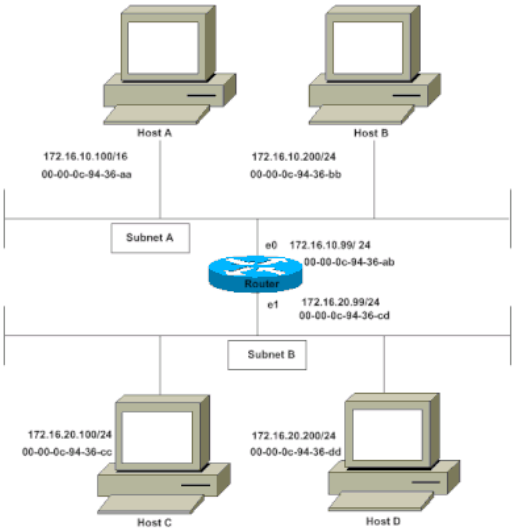
We can see from the SENT line that we (`10.10.14.2`) sent a TCP packet with the `SYN` flag (`S`) to our target (`10.129.2.28`). In the next RCVD line, we can see that the target responds with a TCP packet containing the `RST` and `ACK` flags (`RA`). `RST` and `ACK` flags are used to acknowledge receipt of the TCP packet (`ACK`) and to end the TCP session (`RST`).

Connect Scan

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.

--disable-arp-ping : ARP PROXY

To disable this implicit behavior, use the --disable-arp-ping option. The default behavior is normally faster, but this option is useful on networks using proxy ARP, in which a router speculatively replies to all ARP requests, making every target appear to be up according to ARP scan.



L'hôte A (172.16.10.100) sur le sous-réseau A doit envoyer des paquets à l'hôte D (172.16.20.200) sur le sous-réseau B. Comme l'illustre le schéma, l'hôte A possède un masque de sous-réseau /1
Cela signifie que l'hôte A croit qu'il est directement connecté à tout le réseau 172 16 0 0. Quand un hôte A a besoin de communiquer avec tout périphérique qu'il croit directement connecté, il envoie une requête ARP à la destination. Par conséquent, quand l'hôte A a besoin d'envoyer un paquet à l'hôte D, l'hôte A croit que l'hôte D est directement connecté, ainsi il envoie une requête ARP à l'hôte

Afin d'atteindre l'hôte D (172.16.20.200), l'hôte A a besoin de l'adresse MAC de l'hôte D.

Par conséquent, l'hôte A diffuse une requête ARP sur le sous-réseau A, comme indiqué :

Adresse MAC de l'expéditeur	Adresse IP de l'expéditeur	Adresse MAC cible	Adresse IP cible+F10534
00-00-0c-94-36-aa	172.16.10.100	00-00-00-00-00-00	172.16.20.200

Dans cette requête ARP, l'hôte A (172.16.10.100) demande que l'hôte D envoie (de 172.16.20.200) son adresse MAC. Le paquet de requête ARP est alors encapsulé dans une trame Ethernet avec l'adresse MAC de l'hôte A en tant qu'adresse source et diffusion (FFFF.FFFF.FFFF) comme adresse de destination. Puisque la requête ARP est une diffusion, elle atteint tous les noeuds dans le sous-réseau A, qui inclut l'interface e0 du routeur, mais n'atteint pas l'hôte D. La diffusion n'atteint pas l'hôte D parce que les routeurs, par défaut, Button, ne transfèrent pas des diffusions.

Puisque le routeur sait que l'adresse de destination (172.16.20.200) est sur un autre sous-réseau et peut atteindre l'hôte D, il répond avec sa propre adresse MAC à l'hôte A.

Adresse MAC de l'expéditeur	Adresse IP de l'expéditeur	Adresse MAC cible	Adresse IP cible+F10534
00-00-0c-94-36-ab	172.16.20.200	00-00-0c-94-36-aa	172.16.10.100

C'est la réponse du proxy ARP que le routeur envoie à l'hôte A. Le paquet de réponse ARP du proxy est encapsulé dans une trame Ethernet avec l'adresse MAC du routeur en tant qu'adresse source e l'adresse MAC de l'hôte A en tant qu'adresse de destination. Les réponses ARP sont toujours monodiffusées au demandeur initial.

Dès réception de cette réponse ARP, l'hôte A met à jour sa table ARP, comme indiqué :

Adresse IP	Adresse MAC :
172.16.20.200	00-00-0c-94-36-ab

Dorénavant, l'hôte A renvoie tous les paquets qu'il veut pour accéder à 172.16.20.200 (hôte D) à l'adresse MAC 00-00-0c-94-36-ab (routeur). Puisque le routeur sait comment atteindre l'hôte D, le routeur transfère le paquet à l'hôte D. Le cache ARP sur les hôtes du sous-réseau A est rempli avec l'adresse MAC du routeur pour tous les hôtes du sous-réseau B. Par conséquent, tous les paquets destinés au sous-réseau B sont envoyés au routeur. Le routeur transfère ces paquets aux hôtes dans le sous-réseau B.

Le cache ARP de l'hôte A est indiqué dans cette table :

Adresse IP	Adresse MAC :
172.16.20.200	00-00-0c-94-36-ab
172.16.20.100	00-00-0c-94-36-ab
172.16.10.99	00-00-0c-94-36-ab
172.16.10.200	00-00-0c-94-36-bb

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.

Host and Port Scanning

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -p 139 --packet-trace -n --disable-arp-ping -Pn

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 15:45 CEST
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 <r
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 <m
Nmap scan report for 10.129.2.28
Host is up.

PORT    STATE    SERVICE
139/tcp filtered netbios-ssn
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 2.06 seconds
```

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`).

Let's look at an example of what a UDP scan (`-sU`) can look like and what results it gives us.

UDP Port Scan

Host and Port Scanning

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -F -sU

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:01 CEST
Nmap scan report for 10.129.2.28
Host is up (0.059s latency).
Not shown: 95 closed ports
PORT     STATE         SERVICE
68/udp   open|filtered dhcpc
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
631/udp  open|filtered ipp
5353/udp open          zeroconf
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 98.07 seconds
```

Scanning Options	Description
10.129.2.28	Scans the specified target.
-F	Scans top 100 ports.
-sU	Performs a UDP scan.

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.

Host and Port Scanning

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -sU -Pn -n --disable-arp-ping --packet-trace -p 137 --reason

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:15 CEST
SENT (0.0367s) UDP 10.10.14.2:55478 > 10.129.2.28:137 ttl=57 id=9122 iplen=78
RCVD (0.0398s) UDP 10.129.2.28:137 > 10.10.14.2:55478 ttl=64 id=13222 iplen=257
Nmap scan report for 10.129.2.28
Host is up, received user-set (0.0031s latency).

PORT    STATE SERVICE    REASON
137/udp open  netbios-ns udp-response ttl 64
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

If we get an ICMP response with error code 3 (port unreachable), we know that the port is indeed closed.

Host and Port Scanning

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -sU -Pn -n --disable-arp-ping --packet-trace -p 100 --reason

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:25 CEST
SENT (0.0445s) UDP 10.10.14.2:63825 > 10.129.2.28:100 ttl=57 id=29925 iplen=28
RCVD (0.1498s) ICMP [10.129.2.28 > 10.10.14.2 Port unreachable (type=3/code=3) ] IP [ttl=64 id=11903 iplen=
Nmap scan report for 10.129.2.28
Host is up, received user-set (0.11s latency).

PORT    STATE  SERVICE REASON
100/udp closed unknown port-unreach ttl 64
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in  0.15 seconds
```

For all other ICMP responses, the scanned ports are marked as (**open|filtered**).

Saving the Results

Different Formats

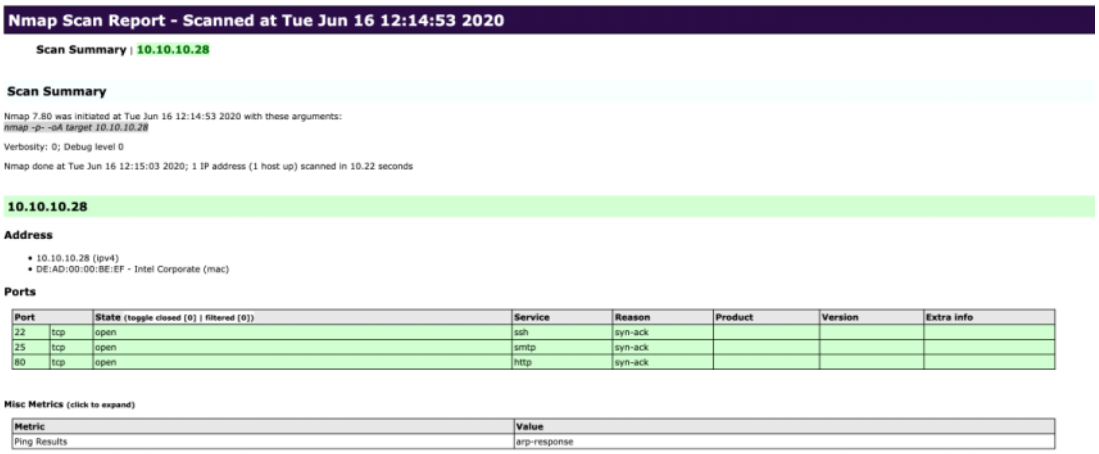
While we run various scans, we should always save the results. We can use these later to examine the differences between the different scanning methods we have used. Nmap can save the results in 3 different formats.

- Normal output (-oN) with the .nmap file extension
- Grepable output (-oG) with the .gnmap file extension
- XML output (-oX) with the .xml file extension

We can also specify the option (-oA) to save the results in all formats. The command could look like this:

Style sheet

```
xsltproc target.xml -o target.html
```



Nmap Scan Report - Scanned at Tue Jun 16 12:14:53 2020

Scan Summary | 10.10.10.28

Scan Summary

Nmap 7.80 was initiated at Tue Jun 16 12:14:53 2020 with these arguments:
nmap -p- -oA target 10.10.10.28

Verbosity: 0; Debug level 0

Nmap done at Tue Jun 16 12:15:03 2020; 1 IP address (1 host up) scanned in 10.22 seconds

10.10.10.28

Address

- 10.10.10.28 (ipv4)
- DE:AD:00:00:BE:EF - Intel Corporate (mac)

Ports

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack			
25	tcp	open	smtp	syn-ack			
80	tcp	open	http	syn-ack			

Misc Metrics (click to expand)

Metric	Value
Ping Results	arp-response

Banner Grabbing

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

Service Enumeration

```
Djerbien@htb[/htb]$ sudo tcpdump -i eth0 host 10.10.14.2 and 10.129.2.28

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Nc

Service Enumeration

```
Djerbien@htb[/htb]$  nc -nv 10.129.2.28 25

Connection to 10.129.2.28 port 25 [tcp/*] succeeded!
220 inlane ESMTP Postfix (Ubuntu)
```

Tcpdump - Intercepted Traffic

Service Enumeration

```
0.14.2.59618 > 10.129.2.28.smtp: Flags [S], seq 1798872233, win 65535, options [mss 1460,nop,wscale 6,nop,nop
29.2.28.smtp > 10.10.14.2.59618: Flags [S.], seq 1130574379, ack 1798872234, win 65160, options [mss 1460,sac
0.14.2.59618 > 10.129.2.28.smtp: Flags [.], ack 1, win 2058, options [nop,nop,TS val 331260304 ecr 1800383922
29.2.28.smtp > 10.10.14.2.59618: Flags [P.], seq 1:36, ack 1, win 510, options [nop,nop,TS val 1800383985 ecr
0.14.2.59618 > 10.129.2.28.smtp: Flags [.], ack 36, win 2058, options [nop,nop,TS val 331260368 ecr 180038398
```

The first three lines show us the three-way handshake.

1. `[SYN]` `18:28:07.128564 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [S], <SNIP>`
2. `[SYN-ACK]` `18:28:07.255151 IP 10.129.2.28.smtp > 10.10.14.2.59618: Flags [S.], <SNIP>`
3. `[ACK]` `18:28:07.255281 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [.], <SNIP>`

After that, the target SMTP server sends us a TCP packet with the `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.

4. `[PSH-ACK]` `18:28:07.319306 IP 10.129.2.28.smtp > 10.10.14.2.59618: Flags [P.], <SNIP>`

The last TCP packet that we sent confirms the receipt of the data with an `ACK`.

5. `[ACK]` `18:28:07.319426 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [.], <SNIP>`

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.
`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.

`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.

We have several ways to define the desired scripts in `Nmap`.

Default Scripts

Nmap Scripting Engine

```
Djerbien@htb[/htb]$ sudo nmap <target> -sC
```

Specific Scripts Category

Nmap Scripting Engine

```
Djerbien@htb[/htb]$ sudo nmap <target> --script <category>
```

Defined Scripts

Nmap Scripting Engine

```
Djerbien@htb[/htb]$ sudo nmap <target> --script <script-name>,<script-name>,...
```

Nmap - Specifying Scripts

Nmap Scripting Engine

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -p 25 --script banner,smtp-commands

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 23:21 CEST
Nmap scan report for 10.129.2.28
Host is up (0.050s latency).

PORT   STATE SERVICE
25/tcp open  smtp
|_banner: 220 inlane ESMTP Postfix (Ubuntu)
|_smtp-commands: inlane, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

Scanning Options	Description
`10.129.2.28`	Scans the specified target.
`-p 25`	Scans only the specified port.
`--script banner,smtp-commands`	Uses specified NSE scripts.

We see that we can recognize the **Ubuntu** distribution of Linux by using the' banner' script. The `smtp-commands` script shows us which commands we can use by interacting with the target SMTP server. In this example, such information may help us to find out existing users on the target. `Nmap` also gives us the ability to scan our target with the aggressive option (`-A`). This scans the target with multiple options as service detection (`-sV`), OS detection (`-O`), traceroute (`--traceroute`), and with the default NSE scripts (`-sC`).

Vulnerability Assessment

Now let us move on to HTTP port 80 and see what information and vulnerabilities we can find using the `vuln` category from `NSE`.

Nmap - Vuln Category

Nmap Scripting Engine

```
Djerbien@htb[/htb]$ sudo nmap 10.129.2.28 -p 80 -sV --script vuln

Nmap scan report for 10.129.2.28
Host is up (0.036s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
| http-enum:
|   /wp-login.php: Possible admin folder
|   /readme.html: Wordpress version: 2
|   /: WordPress version: 5.3.4
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
|   /wp-login.php: Wordpress login page.
|   /wp-admin/upgrade.php: Wordpress login page.
|_  /readme.html: Interesting, a readme.
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-wordpress-users:
| Username found: admin
|_Search stopped at ID #25. Increase the upper limit if necessary with 'http-wordpress-users.limit'
| vulners:
|   cpe:/a:apache:http_server:2.4.29:
|       CVE-2019-0211 7.2  https://vulners.com/cve/CVE-2019-0211
|       CVE-2018-1312 6.8  https://vulners.com/cve/CVE-2018-1312
|       CVE-2017-15715    6.8  https://vulners.com/cve/CVE-2017-15715
<SNIP>
```

Performance

- `-min-parallelism` focuses on the number of concurrent threads or probes used, which affects how many parts of the scan are processed simultaneously.
- `--min-rate` focuses on the speed of packet transmission, aiming to control how quickly packets are sent to the target.

--min-parallelism= 10
--min-rate =1024

Optimized RTT

```
sudo nmap 10.129.2.0/24 -F --initial-rtt-timeout 50ms --max-rtt-timeout 100ms
```

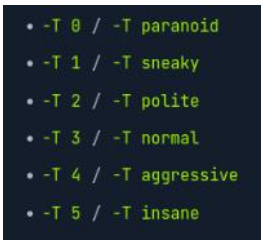
Max Retries

Another way to increase the scans' speed is to specify the **retry rate** of the sent packets (**--max-retries**). **The default value for the retry rate is 10**, so if Nmap does not receive a response for a port, it will not send any more packets to the port and will be skipped.

Rates

```
sudo nmap 10.129.2.0/24 -F -oN tnet.minrate300 --min-rate 300
```

Timing



The default is T3

.