

【PortScan】nmap

常用語法

```
## nmap 預設會scan ip (主機探測) 與 port (服務探測)

## 列出網段內機器的ip (主機探測)
# -sn (no service scan) 不掃描port
# 不要掃描任何連接埠——強制它主要依靠 ICMP 回顯封包 (或本機網路上的 ARP 請求)，
# 如果使用 sudo 運行或直接以 root 使用者身分執行) 來識別目標。除了 ICMP 回顯請求之外，
# 向目標的連接埠 443 發送 TCP SYN 封包，以及向目標的連接埠 80 發送 TCP ACK 封包。

# -n 不使用dns解析
nmap -sn -n 192.168.202.0/24
# -Pn (no ping)掃描之前不需要用ping命令,有些防火牆禁止ping命令。(使用其他方式偵測，速度快)
nmap -Pn -sn -n 192.168.202.0/24
# -oG console output，才能搭配 | 做後續處理
# awk '{prin $2}' 使用awk 取出ip
# > ip.txt 輸出成文字檔
nmap -sn -n 192.168.202.0/24 -oG - | grep Up |awk '{prin $2}' > ips.txt
# cut -d ' ' -f2 => 使用 cut 切出 ip
nmap -sn -n 192.168.202.0/24 -oG - | grep Up |cut -d ' ' -f2 > ips.txt
# -oN {filename} 輸出到檔案
nmap -sn -n 192.168.202.0/24 -oN namp_192.168.202.log

## port 掃描(服務探測)
# namp 預設掃描，常用 1000 port(/usr/share/nmap/nmap-services)
# -p 1-65535 所有port掃描
namp 192.168.203.151 -p 1-65535
# 等於 namp 192.168.203.151 -p -
# 1- : 1以後
# 等於 namp 192.168.203.151 -p 1-
# -65535: 65535 以前
# 等於 namp 192.168.203.151 -p -65535

## 列出ips.txt 主機有開 tcp/25 的主機
# -p port
# -sS syn 掃描 (或是 -sT tcp 連線掃描)
# -iL 從檔案讀取要掃描的主機 (-sL 列出要掃描的ip，只列出不掃描)
# --open 只列出開啟(up)的結果
sudo nmap -p 25 -sS --open 192.168.202.255 -iL ips.txt -oG -
# result
Host: 192.168.202.23 () Status: Up
Host: 192.168.202.23 () Ports: 25/filtered/tcp//smtp//

# 其他參數
-sV 確定開放連接埠上的服務/版本信息
-sV --version-light 嘗試最有可能的探針 (2)
-sV --version-all 嘗試所有可用的探針 (9)
-O 偵測作業系統
--traceroute 運行traceroute到目標
--script=SCRIPTS 要執行的 Nmap 腳本
-sC或者--script=default 運行預設腳本
-A 相當於-sV -O -sC --traceroute
# 輸出
-oN 以正常格式儲存輸出
-oG 以 grepable 格式儲存輸出
-oX 以 XML 格式儲存輸出
-oA 以普通、XML 和 Grepable 格式儲存輸出
```

主機(Ping)(-PX)掃描

```
# host scan
# ARP掃描    sudo nmap -PR -sn MACHINE_IP/24
# ICMP echo掃描    sudo nmap -PE -sn MACHINE_IP/24
# ICMP 時間戳掃描    sudo nmap -PP -sn MACHINE_IP/24
# ICMP 位址遮罩掃描    sudo nmap -PM -sn MACHINE_IP/24
# TCP SYN Ping 掃描    sudo nmap -PS22,80,443 -sn MACHINE_IP/30
# TCP ACK Ping 掃描    sudo nmap -PA22,80,443 -sn MACHINE_IP/30
# UDP Ping 掃描    sudo nmap -PU53,161,162 -sn MACHINE_IP/30
# -n 沒有DNS查找
# -R 所有主機的反向 DNS 查找
# -sn 僅主機發現(不掃描port)
# -Pn 停用主機發現(no ping scan)
# 它的代價是可能需要很長時間才能完成掃描 (如果主機確實死了，那麼 Nmap 仍然會檢查並仔細檢查每個指定的連接埠)

# port 相關option
-p- 所有連接埠
-p1-1023 掃描埠1至1023
-F 100 個最常用端口
-r 按連續順序掃描端口
-T<0-5> -T0最慢，T5最快
--max-rate 50 速率 <= 50 資料包/秒
--min-rate 15 速率 >= 15 資料包/秒
--min-parallelism 100 至少 100 個平行探頭
```

```
# -sL 只列出要掃描的主機(不掃描)
# -n 不進行dns反解
#nmap -n -sL 10.10.12.13/29
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2024-02-28 10:39 GMT
Nmap scan report for 10.10.12.8
Nmap scan report for 10.10.12.9
Nmap scan report for 10.10.12.10
Nmap scan report for 10.10.12.11
Nmap scan report for 10.10.12.12
Nmap scan report for 10.10.12.13
Nmap scan report for 10.10.12.14
Nmap scan report for 10.10.12.15
Nmap done: 8 IP addresses (0 hosts up) scanned in 0.00 seconds
```

arp scan

arp-scan wiki https://www.royhills.co.uk/wiki/index.php/Arp-scan_Documentation

nmap-PR-sn-AttackBox.pcapng			
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help			
arp			
Source	Destination	Protocol	Info
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.1? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.2? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.3? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.4? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.5? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.7? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.8? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.9? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.10? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.11? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.1? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.2? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.3? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.4? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.5? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.7? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.8? Tell 10.10.210.6
nmap-PR-sn-AttackBox.pcapng Packets: 1480 · Displayed: 512 (34.6%) Profile: Default			

```
# -PR 使用arp扫描
# -sn 只扫描ip，不扫描port
$ sudo nmap -PR -sn 10.10.210.6/24
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-02 07:12 BST
Nmap scan report for ip-10-10-210-75.eu-west-1.compute.internal (10.10.210.75)
Host is up (0.00013s latency).
MAC Address: 02:83:75:3A:F2:89 (Unknown)
Nmap scan report for ip-10-10-210-100.eu-west-1.compute.internal (10.10.210.100)
Host is up (-0.100s latency).
MAC Address: 02:63:D0:1B:2D:CD (Unknown)
Nmap scan report for ip-10-10-210-165.eu-west-1.compute.internal (10.10.210.165)
Host is up (0.00025s latency).
MAC Address: 02:59:79:4F:17:B7 (Unknown)
Nmap scan report for ip-10-10-210-6.eu-west-1.compute.internal (10.10.210.6)
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.12 seconds
```

```
# arp-scan
$ sudo arp-scan 10.10.210.6/24
Interface: eth0, datalink type: EN10MB (Ethernet)
WARNING: host part of 10.10.210.6/24 is non-zero
Starting arp-scan 1.9 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
10.10.210.75 02:83:75:3a:f2:89 (Unknown)
10.10.210.100 02:63:d0:1b:2d:cd (Unknown)
10.10.210.165 02:59:79:4f:17:b7 (Unknown)
```

```
4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9: 256 hosts scanned in 2.726 seconds (93.91 hosts/sec). 3 responded
```

icmp scan (-PE)(echo)

icmp 類型 <https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml>

nmap-PE-sn-openvpn.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Echo (ping) request id=0x22e1, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Echo (ping) request id=0xd13b, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Echo (ping) request id=0x65c8, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Echo (ping) request id=0x2b62, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Echo (ping) request id=0x8681, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Echo (ping) request id=0x6a13, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Echo (ping) request id=0x2dbc, seq=0/0, ttl=
10.11.35.214	10.10.68.8	ICMP	Echo (ping) request id=0x2029, seq=0/0, ttl=
10.11.35.214	10.10.68.9	ICMP	Echo (ping) request id=0xf800, seq=0/0, ttl=
10.11.35.214	10.10.68.10	ICMP	Echo (ping) request id=0xca62, seq=0/0, ttl=
10.11.35.214	10.10.68.1	ICMP	Echo (ping) request id=0xe95f, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Echo (ping) request id=0x896e, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Echo (ping) request id=0xdffe, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Echo (ping) request id=0xdf2c, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Echo (ping) request id=0x4602, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Echo (ping) request id=0xd84a, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Echo (ping) request id=0x8ada, seq=0/0, ttl=

nmap-PE-sn-openvpn.pcapng

Packets: 1074 · Displayed: 512 (47.7%) Profile: Default

icmp 類型 <https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml>

-PE 使用icmp 掃描

ICMP 類型 8 Echo -> ICMP 類型 0 Echo Reply 回應

-sn 不進行port掃描

\$ sudo nmap -PE -sn 10.10.68.220/24

Starting Nmap 7.60 (<https://nmap.org>) at 2021-09-02 10:16 BST

Nmap scan report for ip-10-10-68-50.eu-west-1.compute.internal (10.10.68.50)

Host is up (0.00017s latency).

MAC Address: 02:95:36:71:5B:87 (Unknown)

...

Nmap done: 256 IP addresses (8 hosts up) scanned in 2.11 seconds

icmp scan (-PP)(timestamp)

nmap-PP-sn-openvpn.pcapng				
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help				
icmp				
Source	Destination	Protocol	Info	
10.11.35.214	10.10.68.1	ICMP	Timestamp request	id=0xb6bf, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Timestamp request	id=0xcad3, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Timestamp request	id=0x53ce, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Timestamp request	id=0x0149, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Timestamp request	id=0x2ead, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Timestamp request	id=0x3ce5, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Timestamp request	id=0x5de2, seq=0/0, ttl=
10.11.35.214	10.10.68.8	ICMP	Timestamp request	id=0x884d, seq=0/0, ttl=
10.11.35.214	10.10.68.9	ICMP	Timestamp request	id=0xbf35, seq=0/0, ttl=
10.11.35.214	10.10.68.10	ICMP	Timestamp request	id=0x6b44, seq=0/0, ttl=
10.11.35.214	10.10.68.1	ICMP	Timestamp request	id=0x1a28, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Timestamp request	id=0x8586, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Timestamp request	id=0xacce, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Timestamp request	id=0x0cfa, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Timestamp request	id=0xa39f, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Timestamp request	id=0x2279, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Timestamp request	id=0x80cf, seq=0/0, ttl=
nmap-PP-sn-openvpn.pcapng Packets: 1131 · Displayed: 512 (45.3%) Profile: Default				

```
# -PP 使用icmp 時間戳記掃描
# ICMP 類型 13 時間戳記請求 -> ICMP 類型 14 時間戳記請求回應
$ sudo nmap -PP -sn 10.10.68.220/24

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 12:06 EEST
Nmap scan report for 10.10.68.50
Host is up (0.13s latency).
...
Nmap done: 256 IP addresses (8 hosts up) scanned in 10.93 seconds
```

icmp scan (-PM)(address mask)

nmap-PM-sn-openvpn.pcapng			
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help			
icmp			
Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Address mask request id=0xa3c4, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Address mask request id=0xb793, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Address mask request id=0x2d87, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Address mask request id=0x091c, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Address mask request id=0x692c, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Address mask request id=0x4bec, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Address mask request id=0x4d61, seq=0/0, ttl=
10.11.35.214	10.10.68.8	ICMP	Address mask request id=0xb84f, seq=0/0, ttl=
10.11.35.214	10.10.68.9	ICMP	Address mask request id=0x7d19, seq=0/0, ttl=
10.11.35.214	10.10.68.10	ICMP	Address mask request id=0x92be, seq=0/0, ttl=
10.11.35.214	10.10.68.1	ICMP	Address mask request id=0xd204, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Address mask request id=0x683d, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Address mask request id=0x2711, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Address mask request id=0xfde3, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Address mask request id=0x2eb1, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Address mask request id=0x8300, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Address mask request id=0x7400, seq=0/0, ttl=
nmap-PM-sn-openvpn.pcapng Packets: 1178 · Displayed: 512 (43.5%) Profile: Default			

```
# -PM 使用icmp 遮罩掃描
# 位址遮罩查詢Address Mask Request (ICMP 類型 17) 並檢查是否獲得位址遮罩回覆Address Mask Reply (ICMP 類型 18)
$ sudo nmap -PM -sn 10.10.68.220/24
```

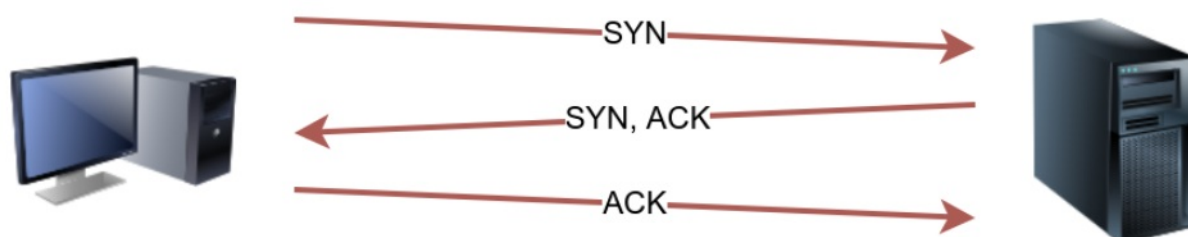
Starting Nmap 7.92 (<https://nmap.org>) at 2021-09-02 12:13 EEST
Nmap done: 256 IP addresses (0 hosts up) scanned in 52.17 seconds

TCP SYN scan (-PS)

```
# TCP SYN 掃描
# -PS 使用tcp sync scan (預設80 port,可設定特殊port 如-PS22,80,8080)
# -sn 不執行port掃描
$ sudo nmap -PS -sn 10.10.68.220/24
```

Starting Nmap 7.92 (<https://nmap.org>) at 2021-09-02 13:45 EEST
Nmap scan report for 10.10.68.52
Host is up (0.10s latency)
...
Nmap done: 256 IP addresses (5 hosts up) scanned in 17.38 seconds

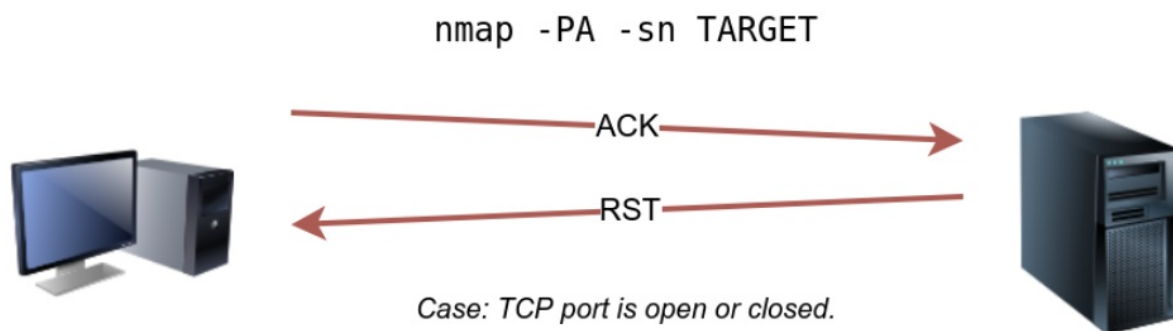
TCP 3-Way Handshake



Case: TCP port is open.

nmap-PS-sn-openvpn.pcapng			
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help			
!openvpn			
Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.2	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.3	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.4	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.5	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.6	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.7	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.8	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.9	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.10	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.1	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.2	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.3	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.4	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.5	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.6	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.7	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
OpenVPN Protocol: Protocol		Packets: 1147 · Displayed: 623 (54.3%) Profile: Default	

TCP **A**CK scan (-PA)



nmap-PA-sn-openvpn.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

!openvpn

Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.2	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.3	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.4	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.5	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.6	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.7	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.8	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.9	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.10	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.1	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.2	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.3	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.4	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.5	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.6	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.7	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0

nmap-PA-sn-openvpn.pcapng

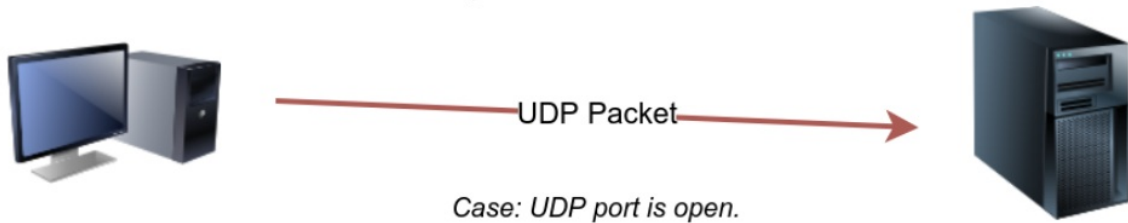
Packets: 1079 · Displayed: 557 (51.6%)

Profile: Default

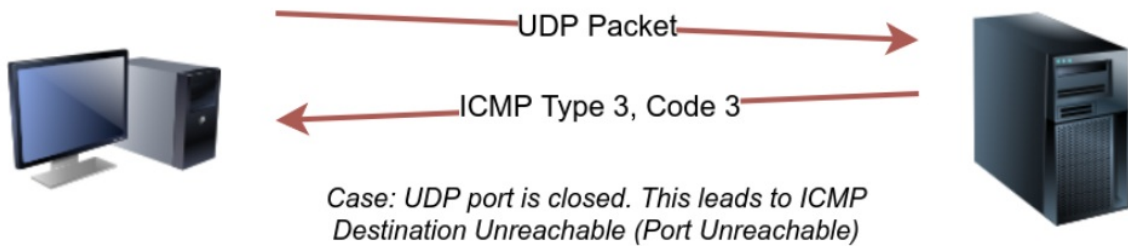
```
# TCP ACK scan
# -PA TCP ACK scan
# -sn 不執行port掃描
# 特權使用者 (root 和 sudoers) 才可以用 ack scan
$ sudo nmap -PA -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 13:46 EEST
Nmap scan report for 10.10.68.52
Host is up (0.11s latency).
...
Nmap done: 256 IP addresses (5 hosts up) scanned in 29.89 seconds
```

UDP scan (-PU)

nmap -PU -sn TARGET



nmap -PU -sn TARGET



nmap-PU-sn-openvpn.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

loopenvpn

Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.2	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.3	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.4	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.5	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.6	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.7	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.8	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.9	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.10	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.1	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.2	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.3	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.4	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.5	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.6	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.7	UDP	57192 → 40125 Len=40

nmap-PU-sn-openvpn.pcapng Packets: 1118 · Displayed: 602 (53.8%) Profile: Default

```
# -PU udp scan
# -sn 不執行port掃描
$ sudo nmap -PU -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 13:45 EEST
Nmap scan report for 10.10.68.52
Host is up (0.10s latency).
...
Nmap done: 256 IP addresses (5 hosts up) scanned in 9.20 seconds
```

dns 反解

```
# -n 不使用dns反解(此選項與下面互斥)
$ sudo nmap -n -sn 192.168.100.10/24

# -R 使用DNS查詢(一般不用加，會使用預設dns)
# --dns-servers {DNS_SERVER} 使用特定dns
```

服務(Service)(-sX) 掃描

我們可以將連接埠分為兩種狀態：

1. 開啟連接埠表示有某個服務正在偵聽該連接埠。
2. 關閉連接埠表示該連接埠沒有服務監聽。

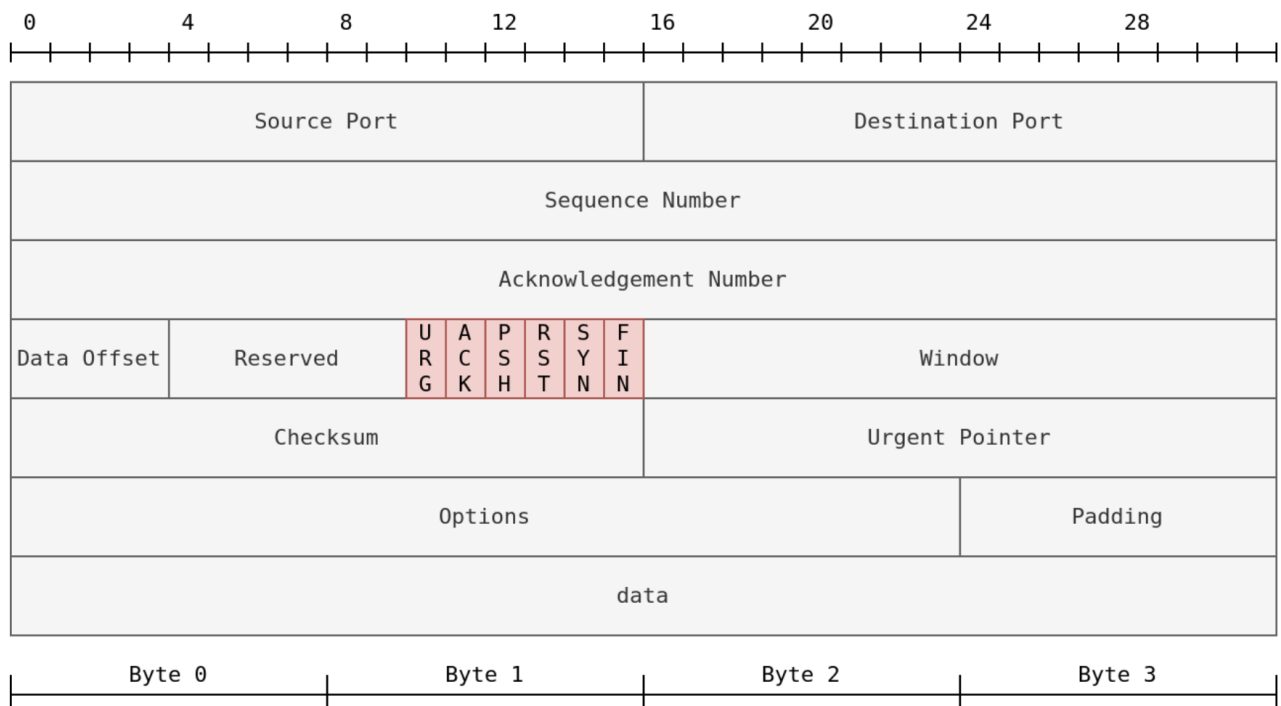
Nmap 六種回應狀態

但在實際情況中，我們需要考慮防火牆的影響。例如，連接埠可能是開放的，但防火牆可能會阻止資料包。因此，Nmap考慮以下六種狀態：

1. **Open**：表示有服務正在監聽指定連接埠。
2. **Closed**：表示沒有服務正在偵聽指定端口，儘管該端口是可訪問的。可存取是指它是可存取的並且不會被防火牆或其他安全設備/程式阻止。
3. **Filtered**：表示Nmap無法確定連接埠是開啟還是關閉，因為該**連接埠無法存取**。這種狀態通常是由於**防火牆阻止** Nmap 到達該連接埠所造成的。Nmap 的封包可能會被阻止到達該連接埠；或者，回應被阻止到達 Nmap 的主機。
4. **Unfiltered**：表示儘管**連接埠可以訪問**，但Nmap無法確定連接埠是開啟還是關閉。使用 **ACK 掃描**時會遇到此狀態 **-sA**。
5. **Open|Filtered**：這意味著Nmap無法確定連接埠是開放的還是已過濾的。
6. **Closed|Filtered**：這表示Nmap無法決定連接埠是關閉還是過濾。

Nmap支援不同類型的 TCP 連接埠掃描。要了解這些連接埠掃描之間的差異，我們需要查看 TCP 標頭。TCP 標頭是 TCP 段的前 24 個位元組。下圖顯示了**RFC 793中定義的 TCP 標頭**。這個人物乍看之下很精緻，但實際上卻很複雜。然而，它很容易理解。在第一行中，我們有來源 TCP 連接埠號碼和目標連接埠號碼。我們可以看到連接埠號碼分配了16位元（2個位元組）。在第二行和第三行中，我們有序號和確認號。每行分配 32 位元（4 位元組），總共 6 行，組成 24 位元組。

TCP Header (RFC793)



TCP 標誌

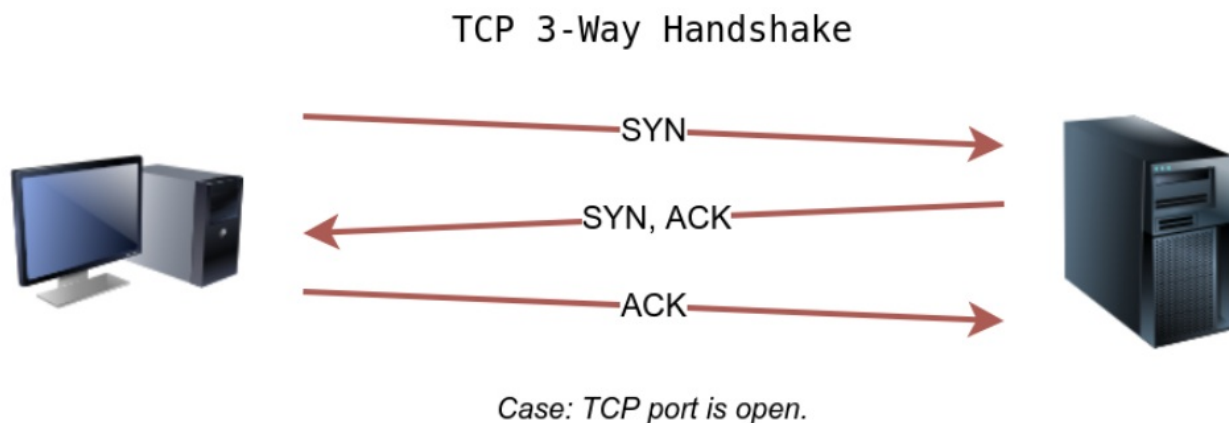
特別是，我們需要關注Nmap可以設定或取消設定的標誌。我們用紅色突出顯示了 TCP 標誌。設定標誌位意味著將其值設為1。從左到右，TCP頭標誌是：

1. **URG**：緊急標誌表示提交的緊急指針是重要的。緊急指針指示傳入資料緊急，並且立即處理設定了 URG 標誌的TCP分段，而無需考慮必須等待先前發送的 TCP 分段。
2. **ACK**：確認標誌表示確認號碼很重要。它用於確認TCP段的接收。
3. **PSH**：推送標誌，要求TCP立即將資料傳遞給應用程式。

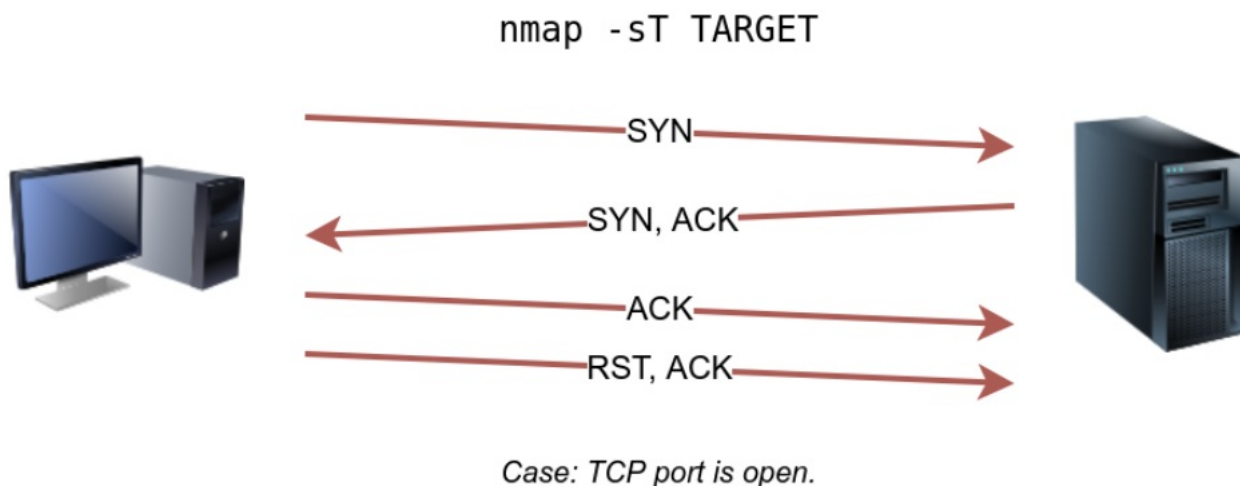
4. **RST**：重設標誌用於重設連線。另一個裝置（例如防火牆）可能會發送它來斷開TCP 連線。當資料傳送到主機且接收端沒有服務來應答時，也會使用此標誌。
5. **SYN**：同步標誌用於啟動TCP 3相交握並與其他主機同步序號。序號應在 TCP 連線建立期間隨機設定。
6. **FIN**：發送方沒有更多資料要傳送。

TCP 連線 掃描 (-sT)

TCP 連線掃描透過完成 TCP 3 次握手來運作。在標準TCP連線建立中，用戶端發送一個設定了SYN標誌的TCP封包，如果連接埠打開，伺服器用SYN/ACK回應；最後，客戶端透過發送ACK完成3次握手。



我們感興趣的是了解 TCP 連接埠是否打開，而不是建立 TCP 連線。因此，一旦透過發送 RST/ACK 確認其狀態，連線就會被中斷。您可以選擇使用執行 TCP 連線掃描



請務必注意，如果您不是特權使用者（root 或 sudoer），則 TCP 連線掃描是發現開放 TCP 連接埠的唯一可能選項。

在下面的 Wireshark 封包擷取視窗中，我們看到 Nmap 會向各種連接埠（256、443、143 等）發送帶有 SYN 標誌的 TCP 封包。預設情況下，Nmap 將嘗試連接到 1000 個最常見的連接埠。關閉的 TCP 連接埠會以 RST/ACK 回應 SYN 封包，以表示它尚未開啟。當我們嘗試與所有關閉的連接埠啟動 TCP 3 向握手時，將重複此模式。

Source	Destination	Protocol	Info
10.10.113.174	10.10.252.27	TCP	40900 → 256 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174	10.10.252.27	TCP	49446 → 443 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174	10.10.252.27	TCP	34466 → 143 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174	10.10.252.27	TCP	50188 → 8888 [SYN] Seq=0 Win=62727 Len=0 MSS=
10.10.113.174	10.10.252.27	TCP	36154 → 5900 [SYN] Seq=0 Win=62727 Len=0 MSS=
10.10.113.174	10.10.252.27	TCP	57134 → 445 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174	10.10.252.27	TCP	40246 → 113 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174	10.10.252.27	TCP	55014 → 111 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174	10.10.252.27	TCP	39962 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=89
10.10.113.174	10.10.252.27	TCP	54696 → 53 [SYN] Seq=0 Win=62727 Len=0 MSS=89
10.10.252.27	10.10.113.174	TCP	443 → 49446 [RST, ACK] Seq=1 Ack=1 Win=0 Len=
10.10.252.27	10.10.113.174	TCP	256 → 40900 [RST, ACK] Seq=1 Ack=1 Win=0 Len=
10.10.252.27	10.10.113.174	TCP	143 → 34466 [SYN, ACK] Seq=0 Ack=1 Win=26847
10.10.113.174	10.10.252.27	TCP	34466 → 143 [ACK] Seq=1 Ack=1 Win=62848 Len=0
10.10.252.27	10.10.113.174	TCP	8888 → 50188 [RST, ACK] Seq=1 Ack=1 Win=0 Len=
10.10.252.27	10.10.113.174	TCP	5900 → 36154 [RST, ACK] Seq=1 Ack=1 Win=0 Len=
10.10.252.27	10.10.113.174	TCP	445 → 57134 [RST, ACK] Seq=1 Ack=1 Win=0 Len=

nmap-sT-AttackBox.pcapng Packets: 2428 · Displayed: 2012 (82.9%) Profile: Default

我們注意到連接埠 143 是開放的，因此它回復了 SYN/ACK，Nmap 透過發送 ACK 完成了 3 次握手。下圖顯示了我們的Nmap主機和目標系統的143埠之間交換的所有資料包。前三個資料包是正在完成的TCP 3次握手。然後，第四個資料包用 RST/ACK 資料包將其撕毀。

Source	Destination	Protocol	Info
10.10.113.174	10.10.252.27	TCP	34466 → 143 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.252.27	10.10.113.174	TCP	143 → 34466 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0
10.10.113.174	10.10.252.27	TCP	34466 → 143 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSv
10.10.113.174	10.10.252.27	TCP	34466 → 143 [RST, ACK] Seq=1 Ack=1 Win=62848 Len=0

```
# -sT tcp連線 scan
# -F 預設是最常用的1000port，使用該參數則為快速掃描(top 100 port)
# -r 預設掃描為亂數取port，該參數可固定排序
$ nmap -sT -F -r 10.10.88.190
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for 10.10.88.190
Host is up (0.0024s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
143/tcp   open  imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.40 seconds
```

TCP SYN 掃描 (-sS)

非特權用戶僅限連線掃描。但是，預設掃描模式是SYN掃描，並且需要特權使用者（root或sudoer）才能運行它。**SYN掃描不需要完成TCP 三向交握**；相反，一旦收到伺服器的回應，它就會斷開連接。因為我們沒有建立 TCP 連接，所以這減少了記錄掃描的機會。我們可以透過使用選項來選擇這種掃描類型 `-sS`。

因為三向握手從未完成，所以信息不會傳遞到應用層，因此**不會出現在任何應用程序日誌**中。SYN掃描也更快速和高效，因為發送和接收的數據包更少。

SYN 掃描有很多好處：

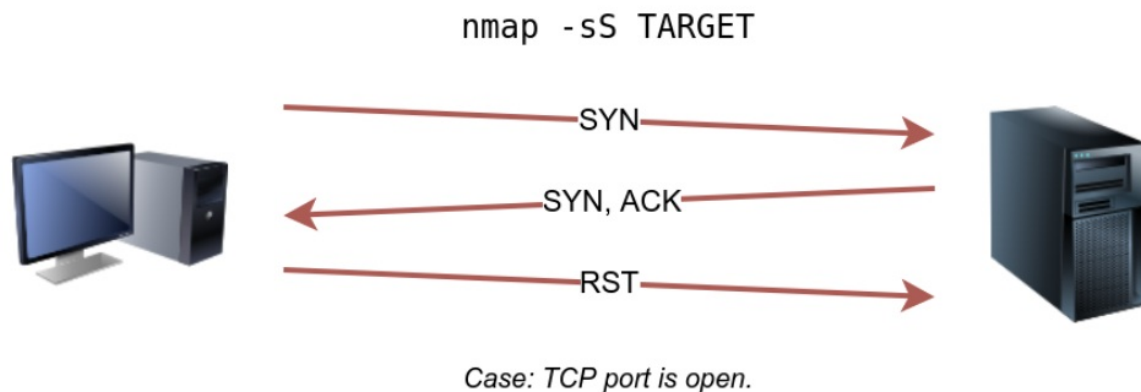
- 它可用於繞過舊的入侵偵測系統，因為它們正在尋找完整的三向握手。現代 IDS 解決方案通常不再出現這種情況；正是由於這個原因，SYN 掃描仍然經常被稱為「隱形」掃描。
- 偵聽開放連接埠的應用程式通常不會記錄 SYN 掃描，因為標準做法是在完全建立連線後記錄連線。這再次體現了 SYN 掃描的隱密性。
- 無需費心為每個連接埠完成（和斷開連接）三向握手，SYN 掃描比標準 TCP 連線掃描快得多。

然而，SYN 掃描有一些缺點，即：

- 它們需要 `sudo` 權限才能在 Linux 中正常運作。這是因為 SYN 掃描需要能夠建立原始資料包（而不是完整的 TCP 握手），預設只有 `root` 使用者才擁有此權限。
- 不穩定的服務有時會因 SYN 掃描而中斷，如果客戶端提供了測試的生產環境，這可能會出現問題。

總而言之，利大於弊。

下圖顯示了在未完成 TCP 三向交握的情況下 TCP SYN 掃描的工作原理。



在下圖的上半部分，我們可以看到一個TCP連接掃描 `-sT` 流量。任何開啟的 TCP 連接埠都需要 Nmap 在關閉連線之前完成 TCP 3 次握手。在下圖的下半部分，我們看到一次SYN掃描如何 `-sS` 不需要完成TCP 3次握手；相反，一旦收到 SYN/ACK 封包，Nmap 就會發送 RST 封包。

nmap-sT-AttackBox.pcapng

Source	Destination	Protocol	Info
10.10.113.174	10.10.252.27	TCP	39962 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SAC
10.10.252.27	10.10.113.174	TCP	80 → 39962 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 M
10.10.113.174	10.10.252.27	TCP	39962 → 80 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=
10.10.113.174	10.10.252.27	TCP	39962 → 80 [RST, ACK] Seq=1 Ack=1 Win=62848 Len=0 T

nmap-sS-AttackBox.pcapng

Source	Destination	Protocol	Info
10.10.113.174	10.10.252.27	TCP	46095 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.252.27	10.10.113.174	TCP	80 → 46095 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0
10.10.113.174	10.10.252.27	TCP	46095 → 80 [RST] Seq=1 Win=0 Len=0

TCP SYN 掃描是以特權使用者身分執行 Nmap、以 root 身分執行或使用 sudo 執行 Nmap 時的**預設掃描模式**

當使用 SYN 掃描來識別關閉和過濾的連接埠時，適用與 TCP 連線掃描(-sT)完全相同的規則。

如果**連接埠關閉**，則伺服器會使用 **RST** TCP 封包進行回應。如果連接埠被防火牆過濾，則 TCP SYN 封包將被丟棄，或透過 TCP 重設進行欺騙。

在這方面，兩次掃描是相同的：最大的區別在於它們處理**開放連接埠**的方式。

```
# -sS tcp ack掃描
$ sudo nmap -sS 10.10.88.190

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for 10.10.88.190
Host is up (0.0073s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open  imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.60 seconds
```

UTP 掃描 (-sU)

與 TCP 不同，UDP 連線是**無狀態**的。這意味著，UDP 連接不是透過來回「握手」來啟動連接，而是依賴將封包發送到目標端口，並本質上希望它們能夠成功。這使得 UDP 非常適合依賴速度而不是品質的連接（例如視訊共享），但是缺乏確認使得 UDP 的掃描變得更加困難（並且速度慢得多）。Nmap UDP 掃描的開關是（-sU）

當資料包發送到開放的 UDP 連接埠時，不應有回應。發生這種情況時，Nmap 將該連接埠稱為 being **open|filtered**。換句話說，它懷疑連接埠是開放的，但可能被防火牆阻止。如果它收到 UDP 回應（這是非常不尋常的），則該連接埠被標記為**open**。更常見的是沒有回應，在這種情況下，會再次發送請求以進行雙重檢查。如果仍然沒有回應，則連接埠被標記為**開啟|已過濾**並且 Nmap 繼續前進。

當封包傳送到**關閉**的UDP 連接埠時，目標應使用 ICMP (ping) 封包進行回應，其中包含該連接埠無法存取的訊息。這清楚地識別了關閉的端口，Nmap 將其標記為關閉的端口並繼續前進。

由於難以識別 UDP 連接埠是否實際打開，因此與各種 TCP 掃描相比，UDP 掃描往往非常緩慢（在連接良好的情況下掃描前 1000 個連

接埠大約需要 20 分鐘）。因此，在 `--top-ports <number>` 啟用的情況下執行 Nmap 掃描通常是一個好習慣。例如，使用 `進行掃描 nmap -sU --top-ports 20 <target>`。將掃描前 20 個最常用的 UDP 端口，從而獲得更可接受的掃描時間。

```
# nmap -sU -p 53 192.168.1.1
# nmap -sU --top-ports 20 <target>
```

掃描範圍與效能

```
# port清單：-p22,80,443 將掃描連接埠 22、80 和 443。
# port範圍：-p1-1023 將掃描 1 到 1023（含）之間的所有port
# 您可以使用 請求掃描所有端口-p-，這將掃描所有 65535 個端口。或是 -p1-65535
# -F 掃描最常見的 100 個port
# --top-ports 10 將檢查10個最常見的連接埠。
```

掃描速度 -T<0-5> (如果您不指定任何計時，Nmap 將使用 normal -T3。)

您可以使用 控制掃描時序-T<0-5>。-T0是最慢的，-T5而是最快的。根據Nmap手冊頁，有六個範本：

為了避免 IDS 警報，您可以考慮-T0或-T1。例如，-T0一次掃描一個端口，並在發送每個探測之間等待 5 分鐘，因此您可以猜測掃描一個目標需要多長時間才能完成。

請注意，-T5 就速度而言，這是最具侵略性的；但是，由於丟包的可能性增加，這可能會影響掃描結果的準確性。

請注意，這-T4通常在 CTF 期間和學習掃描練習目標時使用，而-T1通常在隱密更重要的真實交戰中使用。

控制資料包速率

例如，--max-rate 10或--max-rate=10 確保您的掃描器每秒發送的資料包不超過十個。

--min-rate <number> 控制資料封包速率(最小)

--max-rate <number> 控制資料封包速率(最大)

並行

--min-parallelism <numprobes>此外，您可以使用和 控制探測並行化--max-parallelism <numprobes>。

Nmap 探測目標以發現哪些主機處於活動狀態以及哪些連接埠是開放的；探測並行化指定可以並行運行的此類探測的數量。

例如，--min-parallelism=512推動Nmap保持至少512個探針並行；這512個探測與主機發現和開放連接埠有關。

進階服務掃描

```
# TCP Null Scan  sudo nmap -sN 10.10.156.21
# TCP FIN Scan  sudo nmap -sF 10.10.156.21
# TCP Xmas Scan  sudo nmap -sX 10.10.156.21
# TCP Maimon Scan  sudo nmap -sM 10.10.156.21
# TCP ACK Scan  sudo nmap -sA 10.10.156.21
# TCP Window Scan  sudo nmap -sW 10.10.156.21
# Custom TCP Scan  sudo nmap --scanflags URGACKPSHRSTSYNFIN 10.10.156.21
# Spoofed Source IP  sudo nmap -S SPOOFED_IP 10.10.156.21
# Spoofed MAC Address --spoof-mac SPOOFED_MAC
# Decoy Scan  nmap -D DECOY_IP,ME 10.10.156.21
# Idle (Zombie) Scan sudo nmap -sl ZOMBIE_IP 10.10.156.21
# Fragment IP data into 8 bytes -f
# Fragment IP data into 16 bytes -ff
```

更多參數

--source-port PORT_NUM 指定來源連接埠號

--data-length NUM 附加隨機資料以達到給定長度

--reason 解釋 Nmap 如何得出結論

-v 冗長的

-vv 非常詳細

-d 偵錯

-dd 更多調試細節

NULL、**FIN** 和 **Xmas** TCP 連接埠掃描比我們已經介紹過的其他連接埠掃描不太常用，因此我們不會在這裡深入討論。這三者都是相互關聯的，之所以使用，主要是因為相對而言，它們比 SYN「隱形」掃描更隱密。從 NULL 掃描開始：

- 顧名思義，NULL 掃描（`-sN`）是指發送 TCP 請求時根本沒有設定任何標誌。根據 RFC，如果連接埠關閉，目標主機應使用 **RST** 進行回應。

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	54	36717 → 80 [None] Seq=1 Win=1024 Len=0
2	0.000012387	127.0.0.1	127.0.0.1	TCP	54	80 → 36717 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Acknowledgment number: 0	
Acknowledgment number (raw): 0	
0101 = Header Length: 20 bytes (5)	
▼ Flags: 0x000 (<None>)	
000.	= Reserved: Not set
...0.	= Nonce: Not set
....0... ..	= Congestion Window Reduced (CWR): Not set
.....0... ..	= ECN-Echo: Not set
.....0... ..	= Urgent: Not set
.....0... ..	= Acknowledgment: Not set
.....0... ..	= Push: Not set
.....0... ..	= Reset: Not set
.....0... ..	= Syn: Not set
.....0... ..	= Fin: Not set

- FIN 掃描 (**-sF**) 的運作方式幾乎相同；但是，不是發送完全空的資料包，而是發送帶有 FIN 標誌的請求（通常用於正常關閉活動連線）。如果連接埠關閉，Nmap 再次期望 **RST**。

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	54	33952 → 80 [FIN] Seq=1 Win=1024 Len=0
2	0.000013391	127.0.0.1	127.0.0.1	TCP	54	80 → 33952 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0

Acknowledgment number: 0	
Acknowledgment number (raw): 0	
0101 = Header Length: 20 bytes (5)	
▼ Flags: 0x001 (FIN)	
000.	= Reserved: Not set
...0.	= Nonce: Not set
....0... ..	= Congestion Window Reduced (CWR): Not set
.....0... ..	= ECN-Echo: Not set
.....0... ..	= Urgent: Not set
.....0... ..	= Acknowledgment: Not set
.....0... ..	= Push: Not set
.....0... ..	= Reset: Not set
.....0... ..	= Syn: Not set
.....1... ..	= Fin: Set

- 與此類中的其他兩個掃描一樣，Xmas 掃描 (**-sX**) 會傳送格式錯誤的 TCP 封包，並期望關閉連接埠的 RST 回應。它被稱為**聖誕掃描**，因為當在 Wireshark 中將其視為數據包捕獲時，它設置的標誌 (**PSH**、**URG** 和 **FIN**) 使其看起來像一棵閃爍的聖誕樹。

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	54	46664 → 80 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
2	0.000109904	127.0.0.1	127.0.0.1	TCP	54	80 → 46664 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0

Acknowledgment number: 0	
Acknowledgment number (raw): 0	
0101 = Header Length: 20 bytes (5)	
▼ Flags: 0x029 (FIN, PSH, URG)	
000.	= Reserved: Not set
...0.	= Nonce: Not set
....0... ..	= Congestion Window Reduced (CWR): Not set
.....0... ..	= ECN-Echo: Not set
.....1... ..	= Urgent: Set
.....0... ..	= Acknowledgment: Not set
.....1... ..	= Push: Set
.....0... ..	= Reset: Not set
.....0... ..	= Syn: Not set
.....1... ..	= Fin: Set

這些掃描對開放連接埠的預期回應也是相同的，並且與 UDP 掃描的回應非常相似。如果連接埠打開，則不會對格式錯誤的資料包做出回應。不幸的是（與開放的 UDP 連接埠一樣），如果連接埠受防火牆保護，這也是預期的行為，因此 NULL、FIN 和 Xmas 掃描將僅將連接埠識別為 **open|filtered**、**close** 或 **Filtered**。如果某個連接埠被這些掃描之一識別為已過濾，則通常是因為目標已回應 ICMP 無法到達的封包。

另外值得注意的是，雖然 RFC 793 要求網路主機對關閉連接埠使用 RST TCP 封包回應格式錯誤的封包，而對開放埠完全不回應；實際情況並非總是如此。特別是，眾所周知，**Microsoft Windows**（以及許多 **Cisco** 網路設備）會使用 **RST** 來回應任何格式錯誤的 TCP 封包 - 無論連接埠是否實際開啟。這會導致所有連接埠顯示為關閉。

也就是說，這裡的目標當然是繞過防火牆。許多防火牆配置為將傳入的 TCP 封包丟棄到設定了 SYN 標誌的封鎖連接埠（從而阻止新的連線發起請求）。透過發送不包含 SYN 標誌的請求，我們可以有效地繞過這種防火牆。雖然這在理論上很好，但大多數現代 IDS 解決方案都熟悉這些掃描類型，因此在處理現代系統時不要依賴它們 100% 有效。

作業系統偵測 (-O)

Nmap 可以根據作業系統 (OS) 的行為及其回應中的任何跡象來偵測作業系統 (OS)。可以使用以下命令啟用作業系統偵測 `-O`；這是 OS 中的大寫 `O`。在此範例中，我們 `nmap -sS -O 10.10.85.33` 偵測到作業系統是 Linux 3.X，然後進一步猜測它運行的是 3.13 核心。

```
$ sudo nmap -sS -O 10.10.85.33

Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:04 BST
Nmap scan report for 10.10.85.33
Host is up (0.00099s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open  imap
MAC Address: 02:A0:E7:B5:B6:C5 (Unknown)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3.13
OS details: Linux 3.13
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.91 seconds
```

我們掃描並嘗試偵測其作業系統版本的系統正在執行核心版本 3.16。Nmap 能夠在這種情況下做出接近的猜測。在另一個案例中，我們掃描了核心為 5.13.14 的 Fedora Linux 系統；然而，Nmap 檢測到它是 Linux 2.6.X。好消息是 Nmap 正確偵測到作業系統；不太好的消息是內核版本錯誤。

作業系統偵測非常方便，但影響其準確性的因素很多。首先，Nmap 需要在目標上找到至少一個開放端口和一個關閉端口，才能做出可靠的猜測。此外，由於虛擬化和類似技術的使用不斷增加，來賓作業系統指紋可能會被扭曲。因此，**請始終對作業系統版本持保留態度。**

路由追蹤 (--traceroute)

如果您希望 Nmap 找到您和目標之間的路由器，只需新增 `--traceroute`。在以下範例中，Nmap 將追蹤路由附加到其掃描結果中。`traceroute` 請注意，Nmap 的追蹤路由的工作方式與 Linux 和 macOS 或 `tracert` MS Windows 上的命令略有不同。標準 `traceroute` 從低 TTL（生存時間）資料包開始，並不斷增加，直到到達目標。Nmap 的 `traceroute` 從一個高 TTL 的資料包開始，並不斷減少它。

在下面的範例中，我們 `nmap -sS --traceroute 10.10.85.33` 在 AttackBox 上執行。我們可以看到兩者之間沒有路由器/躍點，因為它們是直接連接的。

```
$ sudo nmap -sS --traceroute 10.10.85.33

Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:05 BST
Nmap scan report for 10.10.85.33
Host is up (0.0015s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open  imap
MAC Address: 02:A0:E7:B5:B6:C5 (Unknown)

TRACEROUTE
HOP RTT    ADDRESS
1  1.48 ms MACHINE_IP

Nmap done: 1 IP address (1 host up) scanned in 1.59 seconds
```


Nmap 腳本引擎 (NSE) (-sC)

使用腳本 NSE

Nmap 提供使用 Lua 語言的腳本的支援。Nmap 腳本引擎 (NSE) 是 Nmap 的一部分，是一個 Lua 解釋器，允許 Nmap 執行用 Lua 語言編寫的 Nmap 腳本。然而，我們不需要學習 Lua 來使用 Nmap 腳本。

`/usr/share/nmap/scripts`，您會注意到有數百個腳本，以它們所針對的協定開頭，方便命名。

您可以指定使用這些已安裝腳本中的任何一個或一組；此外，您可以安裝其他使用者的腳本並將其用於掃描。讓我們從預設腳本開始。 `--script=default` 您可以選擇使用或簡單地新增來執行預設類別中的腳本 `-sC`。除了 `default` 之外，類別還包括 `auth`、`broadcast`、`brute`、`default`、`discovery`、`dos`、`exploit`、`external`、`fuzzer`、`intrusive`、`malware`、`safe`、`version` 和 `vuln`。簡要說明如下表所示。
[可以在此處](#)找到更詳盡的清單。

腳本類別	描述
<code>auth</code>	認證相關腳本
<code>broadcast</code>	透過發送廣播訊息發現主機
<code>brute</code>	對登入執行暴力密碼審核
<code>default</code>	預設腳本，同 <code>-sC</code>
<code>discovery</code>	檢索可存取的信息，例如資料庫表和 DNS 名稱
<code>dos</code>	偵測易受拒絕服務 (DoS) 攻擊的伺服器
<code>exploit</code>	嘗試利用各種易受攻擊的服務
<code>external</code>	使用第三方服務進行檢查，例如 Geoplugin 和 Virustotal
<code>fuzzer</code>	發動模糊攻擊
<code>intrusive</code>	暴力攻擊和利用等侵入性腳本
<code>malware</code>	掃描後門
<code>safe</code>	不會使目標崩潰的安全腳本
<code>version</code>	檢索服務版本
<code>vuln</code>	檢查漏洞或利用易受攻擊的服務

有些腳本屬於多個類別。此外，一些腳本對服務發動暴力攻擊，而另一些腳本則發動 DoS 攻擊並利用系統。因此，如果您不想使服務崩潰或利用它們，那麼在選擇要執行的腳本時要小心，這一點至關重要。

我們使用 Nmap 運行 SYN 掃描 `10.10.218.169` 並在控制台中執行預設腳本，如下所示。命令是 `sudo nmap -sS -sC 10.10.218.169`，其中 `-sC` 將確保 Nmap 將在 SYN 掃描之後執行預設腳本。下面出現了新的詳細資訊。查看 22 埠的 SSH 服務；Nmap 恢復了與正在運行的伺服器相關的所有四個公鑰。考慮另一個例子，連接埠 80 上的 HTTP 服務；Nmap 檢索預設頁面標題。我們可以看到該頁面已保留為預設頁面。

```
# -sC 不帶參數就是跑 預設腳本
$ sudo nmap -sS -sC 10.10.218.169

Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:08 BST
Nmap scan report for ip-10-10-161-170.eu-west-1.compute.internal (10.10.161.170)
Host is up (0.0011s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
| 1024 d5:80:97:a3:a8:3b:57:78:2f:0a:78:ae:ad:34:24:f4 (DSA)
| 2048 aa:66:7a:45:eb:d1:8c:00:e3:12:31:d8:76:8e:ed:3a (RSA)
| 256 3d:82:72:a3:07:49:2e:cb:d9:87:db:08:c6:90:56:65 (ECDSA)
|_ 256 dc:f0:0c:89:70:87:65:ba:52:b1:e9:59:f7:5d:d2:6a (EdDSA)
25/tcp    open  smtp
```

```
|_smtp-commands: debra2.thm.local, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_ssl-cert: Subject: commonName=debra2.thm.local
|_Not valid before: 2021-08-10T12:10:58
|_Not valid after: 2031-08-08T12:10:58
|_ssl-date: TLS randomness does not represent time
80/tcp open  http
|_http-title: Welcome to nginx on Debian!
110/tcp open  pop3
|_pop3-capabilities: RESP-CODES CAPA TOP SASL UIDL PIPELINING AUTH-RESP-CODE
111/tcp open  rpcbind
|_rpcinfo:
|_program version  port/proto  service
|_100000 2,3,4      111/tcp  rpcbind
|_100000 2,3,4      111/udp  rpcbind
|_100024 1          38099/tcp status
|_100024 1          54067/udp status
143/tcp open  imap
|_imap-capabilities: LITERAL+ capabilities IMAP4rev1 OK Pre-login ENABLE have LOGINDISABLEDA0001 listed SASL-IR ID more post-
login LOGIN-REFERRALS IDLE
MAC Address: 02:A0:E7:B5:B6:C5 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 2.21 seconds
```



要運行特定腳本，我們將使用 `--script=<script-name>`，例如 `--script=http-fileupload-exploiter`。也可以使用名稱或模式（例如 `--script "ftp*"`，其中包括）來指定腳本 `ftp-brute`。如果您不確定腳本的作用，可以使用文字閱讀器（例如）`less` 或文字編輯器開啟腳本檔案。在的例子中 `ftp-brute`，它指出：“對 FTP 伺服器執行暴力密碼審核。”您必須小心，因為某些腳本非常具有侵入性。此外，某些腳本可能適用於特定伺服器，如果隨機選擇，則會浪費您的時間而沒有任何好處。與往常一樣，請確保您有權在目標伺服器上啟動此類測試。

透過用逗號分隔多個腳本可以以這種方式同時運行。例如：`--script=smb-enum-users,smb-enum-shares`。

某些腳本需要參數（例如，憑證，如果它們正在利用經過身份驗證的漏洞）。這些可以透過 Nmap 開關給出 `--script-args`。腳本就是一個例子 `http-put`（用於使用 PUT 方法上傳檔案）。這需要兩個參數：將檔案上傳到的 URL 以及檔案在磁碟上的位置。例如：

```
nmap -p 80 --script http-put --script-args http-put.url='/dav/shell.php',http-put.file='./shell.php'
```

請注意，參數之間以逗號分隔，並用句點（即 `<script-name>.<argument>`）連接到對應的腳本。

```
# --script=<script-name> 運行特定腳本
# 透過用逗號分隔多個腳本可以以這種方式同時運行 --script=smb-enum-users,smb-enum-shares
# --script-args 給定參數
# 參數之間以逗號分隔，並用句點（即 <script-name>.<argument>）連接到對應的腳本
$ nmap -p 80 --script http-put --script-args http-put.url='/dav/shell.php',http-put.file='./shell.php'
```

可以在此處找到腳本及其相應參數（以及範例用例）的完整清單。

讓我們考慮一個良性腳本，`http-date` 我們猜測它會檢索 http 伺服器日期和時間，這在其描述中確實得到了證實：「從類似 HTTP 的服務獲取日期。此外，它還會列印日期與當地時間的差異...」，我們執行 `sudo nmap -sS -n --script "http-date" 10.10.218.169` 如下控制台所示。

```
# --script=<script-name> 運行特定腳本
# 透過用逗號分隔多個腳本可以以這種方式同時運行 --script=smb-enum-users,smb-enum-shares
# --script-args 給定參數
# 參數之間以逗號分隔，並用句點（即 <script-name>.<argument>）連接到對應的腳本
# nmap -p 80 --script http-put --script-args http-put.url='/dav/shell.php',http-put.file='./shell.php'

$ sudo nmap -sS -n --script "http-date" 10.10.218.169

Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 08:04 BST
Nmap scan report for 10.10.218.169
Host is up (0.0011s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
|_http-date: Fri, 10 Sep 2021 07:04:26 GMT; 0s from local time.
110/tcp   open  pop3
111/tcp   open  rpcbind
```

```
143/tcp open  imap
MAC Address: 02:44:87:82:AC:83 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.78 seconds
```

尋找腳本

我們有兩種選擇，最好將它們結合使用。

第一個是Nmap 網站上的頁面，其中包含所有官方腳本的清單。

第二個是攻擊機器上的本機儲存。Nmap 在 Linux 上將其腳本儲存在 `/usr/share/nmap/scripts`。預設情況下，所有 NSE 腳本都儲存在該目錄中—當您指定腳本時，Nmap 會在該目錄中尋找腳本。

有兩種方法可以搜尋已安裝的腳本。一種是使用 `/usr/share/nmap/scripts/script.db` 文件。儘管有擴展名，但這實際上並不是一個資料庫，而是一個包含每個可用腳本的檔案名稱和類別的格式化文字檔案。

```
# file /usr/share/nmap/scripts/script.db
/usr/share/nmap/scripts/script.db: ASCII text

# head /usr/share/nmap/scripts/script.db
Entry { filename = "acarsd-info.nse", categories = { "discovery", "safe", } }
Entry { filename = "address-info.nse", categories = { "default", "safe", } }
Entry { filename = "afp-brute.nse", categories = { "brute", "intrusive", } }
Entry { filename = "afp-ls.nse", categories = { "discovery", "safe", } }
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln", } }
Entry { filename = "afp-serverinfo.nse", categories = { "default", "discovery", "safe", } }
Entry { filename = "afp-showmount.nse", categories = { "discovery", "safe", } }
Entry { filename = "ajp-auth.nse", categories = { "auth", "default", "safe", } }
Entry { filename = "ajp-brute.nse", categories = { "brute", "intrusive", } }
Entry { filename = "ajp-headers.nse", categories = { "discovery", "safe", } }
```

Nmap 使用此文件來追蹤（並利用）腳本引擎的腳本；但是，我們也可以透過 `grep`，或是 `ls` 來找出腳本。例如：

```
# grep "ftp" /usr/share/nmap/scripts/script.db
Entry { filename = "ftp-anon.nse", categories = { "auth", "default", "safe", } }
Entry { filename = "ftp-bounce.nse", categories = { "default", "safe", } }
Entry { filename = "ftp-brute.nse", categories = { "brute", "intrusive", } }
Entry { filename = "ftp-libopie.nse", categories = { "intrusive", "vuln", } }
Entry { filename = "ftp-proftpd-backdoor.nse", categories = { "exploit", "intrusive", "malware", "vuln", } }
Entry { filename = "ftp-syst.nse", categories = { "default", "discovery", "safe", } }
Entry { filename = "ftp-vsftpd-backdoor.nse", categories = { "exploit", "intrusive", "malware", "vuln", } }
Entry { filename = "ftp-vuln-cve2010-4221.nse", categories = { "intrusive", "vuln", } }
Entry { filename = "tftp-enum.nse", categories = { "discovery", "intrusive", } }

# ls -al /usr/share/nmap/scripts/*ftp*
-rw-r--r-- 1 root root 4564 Apr 16 2018 /usr/share/nmap/scripts/ftp-anon.nse
-rw-r--r-- 1 root root 3253 Apr 16 2018 /usr/share/nmap/scripts/ftp-bounce.nse
-rw-r--r-- 1 root root 3108 Apr 16 2018 /usr/share/nmap/scripts/ftp-brute.nse
-rw-r--r-- 1 root root 3258 Apr 16 2018 /usr/share/nmap/scripts/ftp-libopie.nse
-rw-r--r-- 1 root root 3295 Apr 16 2018 /usr/share/nmap/scripts/ftp-proftpd-backdoor.nse
-rw-r--r-- 1 root root 3810 Apr 16 2018 /usr/share/nmap/scripts/ftp-syst.nse
-rw-r--r-- 1 root root 6007 Apr 16 2018 /usr/share/nmap/scripts/ftp-vsftpd-backdoor.nse
-rw-r--r-- 1 root root 5943 Apr 16 2018 /usr/share/nmap/scripts/ftp-vuln-cve2010-4221.nse
-rw-r--r-- 1 root root 5678 Apr 16 2018 /usr/share/nmap/scripts/tftp-enum.nse
```

安裝腳本

```
# 直接更新
sudo apt update && sudo apt install nmap

# 手動下載
sudo wget -O /usr/share/nmap/scripts/<script-name>.nse https://svn.nmap.org/nmap/scripts/<script-name>.nse
# 下載完更新DB
nmap --script-updatedb
```

防火牆規避

您的典型 Windows 主機將使用其預設防火牆阻止所有 ICMP 封包。這就帶來了一個問題：我們不僅經常使用ping來手動建立目標的活動，Nmap 預設也會做同樣的事情。這表示 Nmap 會將具有此防火牆配置的主機註冊為死亡主機，並且根本不會掃描它。

因此，我們需要一種方法來繞過這個配置。幸運的是，Nmap 為此提供了一個選項：`-Pn`，它告訴 Nmap 在掃描主機之前不要費心 ping 主機。這意味著 Nmap 將始終將目標主機視為活動的，從而有效地繞過 ICMP 區塊；然而，它的代價是可能需要很長時間才能完成掃描（如果主機確實死了，那麼 Nmap 仍然會檢查並仔細檢查每個指定的連接埠）。

值得注意的是，如果您已經直接位於本機網路上，Nmap 也可以使用 ARP 請求來確定主機活動。

Nmap 認為還有多種其他開關可用於規避防火牆。我們不會詳細介紹這些內容，但是，您可以在[在這裡](#)找到它們。

以下開關尤其值得注意：

- `-f`:- 用於對資料包進行分段（即，將它們分成更小的部分），從而降低防火牆或 IDS 檢測到資料包的可能性。
- `-f` 的替代方案，但提供對資料包大小的更多控制：`--mtu <number>`，接受用於傳送的資料包的最大傳輸單元大小。這必須是 8 的倍數。
- `--scan-delay <time>ms`:- 用於在發送的資料包之間添加延遲。如果網路不穩定，這非常有用，而且還可以逃避任何可能存在的基於時間的防火牆/IDS 觸發器。
- `--badsum`:- 這用於產生資料包的無效校驗和。任何真正的 TCP/IP 堆疊都會丟棄此封包，但是，防火牆可能會自動回應，而無需檢查封包的校驗和。因此，此開關可用於確定防火牆/IDS 的存在。

Nmap 參數說明

以下是Nmap 7.94的說明及參數翻譯：

Nmap 7.94 (<https://nmap.org>)

用法：nmap [掃描類型] [選項] {目標指定}

目標指定：

可傳遞主機名、IP地址、網絡等等。

例如：scanme.nmap.org、microsoft.com/24、192.168.0.1; 10.0.0-255.1-254

`-iL` <輸入文件名>：從主機/網絡列表輸入

`-iR` <主機數量>：選擇隨機目標

`--exclude` <主機1[,主機2[,主機3],...>：排除主機/網絡

`--excludefile` <排除文件>：從文件排除列表

主機發現：

`-sL`：列表掃描 - 簡單列出要掃描的目標

`-sn`：Ping掃描 - 停用端口掃描

`-Pn`：將所有主機視為在線 - 跳過主機發現

`-PS/PA/PU/PY`<端口列表>：TCP SYN/ACK、UDP或SCTP發現到指定端口

`-PE/PP/PM`：ICMP echo、timestamp和netmask請求發現探針

`-PO`[協議列表]：IP協議Ping

`-n/-R`：永不執行DNS解析/總是解析 [默認：有時]

`--dns-servers` <伺服器1[,伺服器2],...>：指定自定義DNS伺服器

`--system-dns`：使用作業系統的DNS解析器

`--traceroute`：追蹤到每個主機的跳躍路徑

掃描技術：

`-sS/sT/sA/sW/sM`：TCP SYN/Connect()/ACK/Window/Maimon掃描

`-sU`：UDP掃描

`-sN/sF/sX`：TCP Null、FIN和Xmas掃描

`--scanflags` <標誌>：自定義TCP掃描標誌

`-sI` <殭屍主機[:探針端口]>：閒置掃描

`-sY/sZ`：SCTP INIT/COOKIE-ECHO掃描

`-sO`：IP協議掃描

`-b` <FTP中繼主機>：FTP彈跳掃描

端口指定和掃描順序：

`-p` <端口範圍>：僅掃描指定的端口

例如：`-p22`；`-p1-65535`；`-p U:53,111,137,T:21-25,80,139,8080,S:9`

`--exclude-ports` <端口範圍>：排除掃描指定的端口

`-F`：快速模式 - 掃描比默認更少的端口

`-r`：按順序掃描端口 - 不隨機化

`--top-ports` <數量>：掃描最常見的<數量>個端口

`--port-ratio` <比率>：掃描比<比率>常見的端口

服務/版本檢測：

`-sV`：探測打開的端口以確定服務/版本信息

`--version-intensity` <級別>：設置從0（輕）到9（嘗試所有探針）的級別

`--version-light`：限制最可能的探針（級別2）

`--version-all`：嘗試每個單獨的探針（級別9）

`--version-trace`：顯示詳細的版本掃描活動（用於調試）

腳本掃描：

--sC：等同於 --script=default
--script=<Lua腳本>：<Lua腳本> 是目錄、腳本文件或腳本類別的逗號分隔列表
--script-args=<n1=v1,[

n2=v2,...]>：提供給腳本的參數

--script-args-file=文件名：在文件中提供NSE腳本參數
--script-trace：顯示發送和接收的所有數據
--script-updatedb：更新腳本數據庫
--script-help=<Lua腳本>：顯示有關腳本的幫助
<Lua腳本> 是腳本文件或腳本類別的逗號分隔列表

OS檢測：

-O：啟用OS檢測
--osscan-limit：限制OS檢測對有希望的目標
--osscan-guess：更積極地猜測OS

時間和性能：

需要 <時間> 的選項以秒為單位，或在值后附加 'ms'（毫秒）、's'（秒）、'm'（分鐘）或'h'（小時）（例如 30m）。
-T<0-5>：設置時間模板（較高表示更快）
--min-hostgroup/max-hostgroup <大小>：平行主機掃描組大小
--min-parallelism/max-parallelism <探針數量>：探針並行
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <時間>：指定探針往返時間
--max-retries <嘗試次數>：限制端口掃描探針的重發次數
--host-timeout <時間>：在此時間后放棄對目標的掃描
--scan-delay/--max-scan-delay <時間>：調整探針之間的延遲
--min-rate <數量>：每秒發送的封包數不低於<數量>
--max-rate <數量>：每秒發送的封包數不高於<數量>

防火牆/入侵檢測和欺騙：

-f; --mtu <值>：分段封包（可選附帶MTU）
-D <欺騙主機1,欺騙主機2[,自己],...>：用欺騙主機偽裝掃描
-S <IP地址>：偽造源地址
-e <接口>：使用指定的接口
-g/--source-port <端口號>：使用指定的端口號
--proxies <URL1,[URL2],...>：通過HTTP/SOCKS4代理中繼連接
--data <十六進制字符串>：附加自定義有效載荷到發送的封包
--data-string <字符串>：附加自定義ASCII字符串到發送的封包
--data-length <數量>：附加隨機數據到發送的封包
--ip-options <選項>：發送指定IP選項的封包
--ttl <值>：設置IP生存時間字段
--spoof-mac <MAC地址/前綴/供應商名稱>：偽造MAC地址
--badsum：發送具有虛偽的TCP/UDP/SCTP校驗和的封包

輸出：

-oN/-oX/-oS/-oG <文件>：將掃描輸出到正常、XML、s|<rlpt klddi3和Grepable格式，分別到指定的文件名
-oA <基本名>：一次以三種主要格式輸出
-v：增加詳細程度（使用 -vv 或更多以獲得更大效果）
-d：增加調試程度（使用 -dd 或更多以獲得更大效果）
--reason：顯示端口處於特定狀態的原因
--open：僅顯示打開（或可能打開）的端口
--packet-trace：顯示發送和接收的所有封包
--iflist：打印主機接口和路由（用於調試）
--append-output：附加到指定的輸出文件而不是覆蓋它
--resume <文件名>：恢復中斷的掃描
--noninteractive：禁用通過鍵盤的運行時交互
--stylesheet <路徑/URL>：XSL樣式表以將XML輸出轉換為HTML
--webxml：從Nmap.Org引用樣式表以實現更具可移植性的XML
--no-stylesheet：防止將XSL樣式表與XML輸出關聯

其他：

-6：啟用IPv6掃描
-A：啟用OS檢測、版本檢測、腳本掃描和路由跟蹤
--datadir <目錄名>：指定自定義Nmap數據文件位置
--send-eth/--send-ip：使用原始以太網幀或IP封包發送
--privileged：假定用戶具有完全特權
--unprivileged：假定用戶缺乏原始套接字特權
-V：打印版本號
-h：打印此幫助摘要頁。

範例：

nmap -v -A scanme.nmap.org
nmap -v -sn 192.168.0.0/16 10.0.0.0/8
nmap -v -iR 10000 -Pn -p 80

其他掃描

SMB 掃描

NetBIOS服務聽取TCP端口139，以及多個UDP端口。需要注意的是，SMB（TCP端口445）和NetBIOS是兩個獨立的協議。NetBIOS是一個獨立的會話層協議和服務，允許本地網絡上的計算機彼此通信。雖然現代SMB的實現可以在沒有NetBIOS的情況下運行，但為了向後兼容性，通常一起啟用NetBIOS over TCP（NBT）。這也意味著這兩個服務的列舉通常是相互關聯的。可以使用類似以下語法的工具，如nmap，來掃描這些服務：

```
kali@kali:~$ nmap -v -p 139,445 -oG smb.txt 192.168.50.1-254

kali@kali:~$ cat smb.txt
# Nmap 7.92 scan initiated Thu Mar 17 06:03:12 2022 as: nmap -v -p 139,445 -oG smb.txt 192.168.50.1-254
# Ports scanned: TCP(2;139,445) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 192.168.50.1 () Status: Down
...
Host: 192.168.50.21 () Status: Up
Host: 192.168.50.21 () Ports: 139/closed/tcp//netbios-ssn///, 445/closed/tcp//microsoft-ds///
...
Host: 192.168.50.217 () Status: Up
Host: 192.168.50.217 () Ports: 139/closed/tcp//netbios-ssn///, 445/closed/tcp//microsoft-ds///
# Nmap done at Thu Mar 17 06:03:18 2022 -- 254 IP addresses (15 hosts up) scanned in 6.17 seconds
```

Nmap還提供了許多有用的NSE腳本，我們可以使用它們來發現和列舉SMB服務。這些腳本位於/usr/share/nmap/scripts目錄中。

```
kali@kali:~$ ls -l /usr/share/nmap/scripts/smb*
/usr/share/nmap/scripts/smb2-capabilities.nse
/usr/share/nmap/scripts/smb2-security-mode.nse
/usr/share/nmap/scripts/smb2-time.nse
/usr/share/nmap/scripts/smb2-vuln-uptime.nse
/usr/share/nmap/scripts/smb-brute.nse
/usr/share/nmap/scripts/smb-double-pulsar-backdoor.nse
/usr/share/nmap/scripts/smb-enum-domains.nse
/usr/share/nmap/scripts/smb-enum-groups.nse
/usr/share/nmap/scripts/smb-enum-processes.nse
/usr/share/nmap/scripts/smb-enum-sessions.nse
/usr/share/nmap/scripts/smb-enum-shares.nse
/usr/share/nmap/scripts/smb-enum-users.nse
/usr/share/nmap/scripts/smb-os-discovery.nse
...
```

我們找到了一些有趣的Nmap SMB NSE腳本，可以通過SMB執行各種任務，如OS發現和列舉。

SMB發現腳本僅在目標上啟用了SMBv1時才能工作，這在現代Windows版本中不是默認情況。然而，仍然有很多遺留系統運行著SMBv1，我們已經在Windows主機上啟用了這個特定版本，以模擬這種情況。

讓我們在Windows 11客戶端上嘗試smb-os-discovery模塊。

```
kali@kali:~$ nmap -v -p 139,445 --script smb-os-discovery 192.168.50.152
...
PORT      STATE SERVICE  REASON
139/tcp   open  netbios-ssn syn-ack
445/tcp   open  microsoft-ds syn-ack

Host script results:
| smb-os-discovery:
|   OS: Windows 10 Pro 22000 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: client01
|   NetBIOS computer name: CLIENT01\x00
|   Domain name: megacorpstwo.com
|   Forest name: megacorpstwo.com
|   FQDN: client01.megacorpstwo.com
|_  System time: 2022-03-17T11:54:20-07:00
```

...

SNMP 掃描

要掃描開放的SNMP端口，我們可以運行nmap，使用-sU選項進行UDP掃描，並使用--open選項限制輸出，僅顯示已打開的端口。這將幫助我們識別正在運行SNMP服務的設備。

```
kali@kali:~$ sudo nmap -sU --open -p 161 192.168.50.1-254 -oG open-snmp.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-14 06:02 EDT
Nmap scan report for 192.168.50.151
Host is up (0.10s latency).

PORT      STATE SERVICE
161/udp    open  snmp

Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds
...
```

namp.sh

```
#!/bin/bash

# 檢查是否提供了 IP 地址作為參數
if [ -z "$1" ]; then
    echo "用法: $0 <target_ip> [output_file]"
    exit 1
fi

TARGET_IP=$1
OUTPUT_FILE=$2

# 執行全端口掃描，將結果保存到臨時文件中
echo "開始全端口掃描目標 $TARGET_IP ..."
sudo nmap -p- $TARGET_IP -oN all_ports_scan.txt

# 提取開放端口
echo "提取開放端口..."
open_ports=$(grep "open" all_ports_scan.txt | awk -F/ '{print $1}' | tr '\n' ',' | sed 's/,,$//')

# 檢查是否有開放端口
if [ -z "$open_ports" ]; then
    echo "沒有找到開放端口。"
    rm -f all_ports_scan.txt
    exit 1
fi

echo "開放端口: $open_ports"

# 構建詳細掃描命令
SCAN_COMMAND="sudo nmap -A -p $open_ports $TARGET_IP"

# 執行詳細掃描並處理輸出
if [ -n "$OUTPUT_FILE" ]; then
    echo "開始詳細掃描開放端口 $open_ports，並將結果輸出到 $OUTPUT_FILE ..."
    $SCAN_COMMAND -oN "$OUTPUT_FILE"
else
    echo "開始詳細掃描開放端口 $open_ports ..."
    $SCAN_COMMAND
fi

# 清理臨時文件
rm -f all_ports_scan.txt

echo "掃描完成。"
```

🔄修訂版本 #58

★由 treeman 建立於 6 🍈Q🍈@🍈🍈 2023 02:58:06

✎由 treeman 更新於 2 🍈K🍈🍈 2024 18:34:28