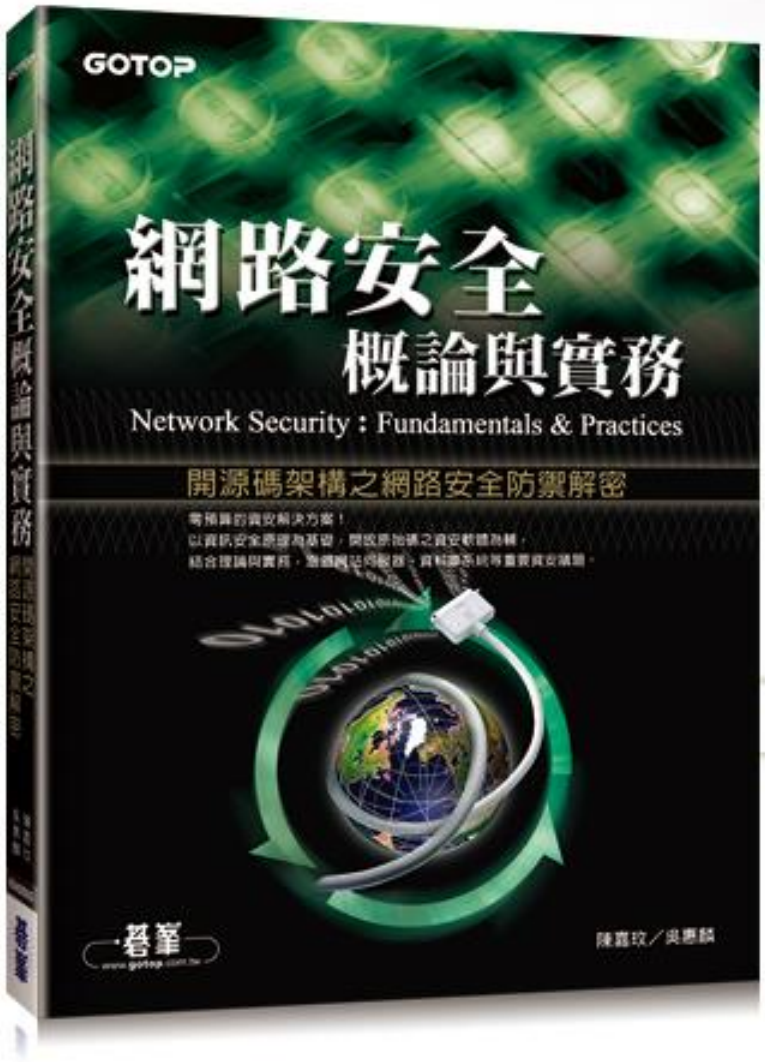


網路安全
概論與實務

Network Security : Fundamentals & Practices

開源碼架構之網路安全防禦解密

CHAPTER

2 埠掃描偵測

2.1 本章目的

所有犯罪行為通常都具有一定的行為模式，即確立目標 → 勘察環境 → 決定適當的工具 → 攻擊。網路攻擊行為自然也不例外，當駭客決定目標後，即會以掃描工具來針對目標進行掃描，而後再根據掃描結果決定攻擊方式來進行攻擊。因此，如果能儘早得知進行掃描的惡意來源 IP，即可在惡意攻擊者進行下一步攻擊時，先行防堵，進而降低系統的危害。而在

2.1 本章目的

本單元中，我們將介紹一套開源碼社群中的掃描偵測工具（Scanlogd），用來偵測實施埠掃描的惡意來源 IP，以便系統管理者能即時偵測埠掃描相關事件，來採取必要的措施。

回首頁

2.2 實作設備

套件名稱	官方網址	說明
scanlogd	http://www.openwall.com/scanlogd/	埠掃描偵測工具（Port Scan Attack detector）

回首頁

2.3 背景說明

在駭客決定攻擊目標後，第一個步驟往往是先行探測目標主機的作業系統。由於每一家作業系統對於封包的處理方式均不同，探測軟體即可利用此種特性來辨識出不同的作業系統。由於此種方式就像人類的指紋一樣，所以又稱為「作業系統指紋（OS Fingerprint）」辨識，作業系統探測方式可分為主動式及被動式。如下所述：

2.3 背景說明

作業系統指紋（OS Fingerprint）

- 主動式：
  - 傳送特別打造的 TCP、UDP 或 ICMP 封包至被探測主機
  - 根據被探測主機回傳的封包特徵，來辨識出該主機的作業系統
  - 此種工具以 nmap 為代表。官方網址為：http://nmap.org/。
- 被動式：
  - 利用 sniffer（嗅探）技術來監看往來封包
  - 藉由往來封包的特徵來辨識出作業系統
  - 此種探測軟體以 p0f 為代表。官方網址為 http://lcamtuf.coredump.cx/p0f.shtml。

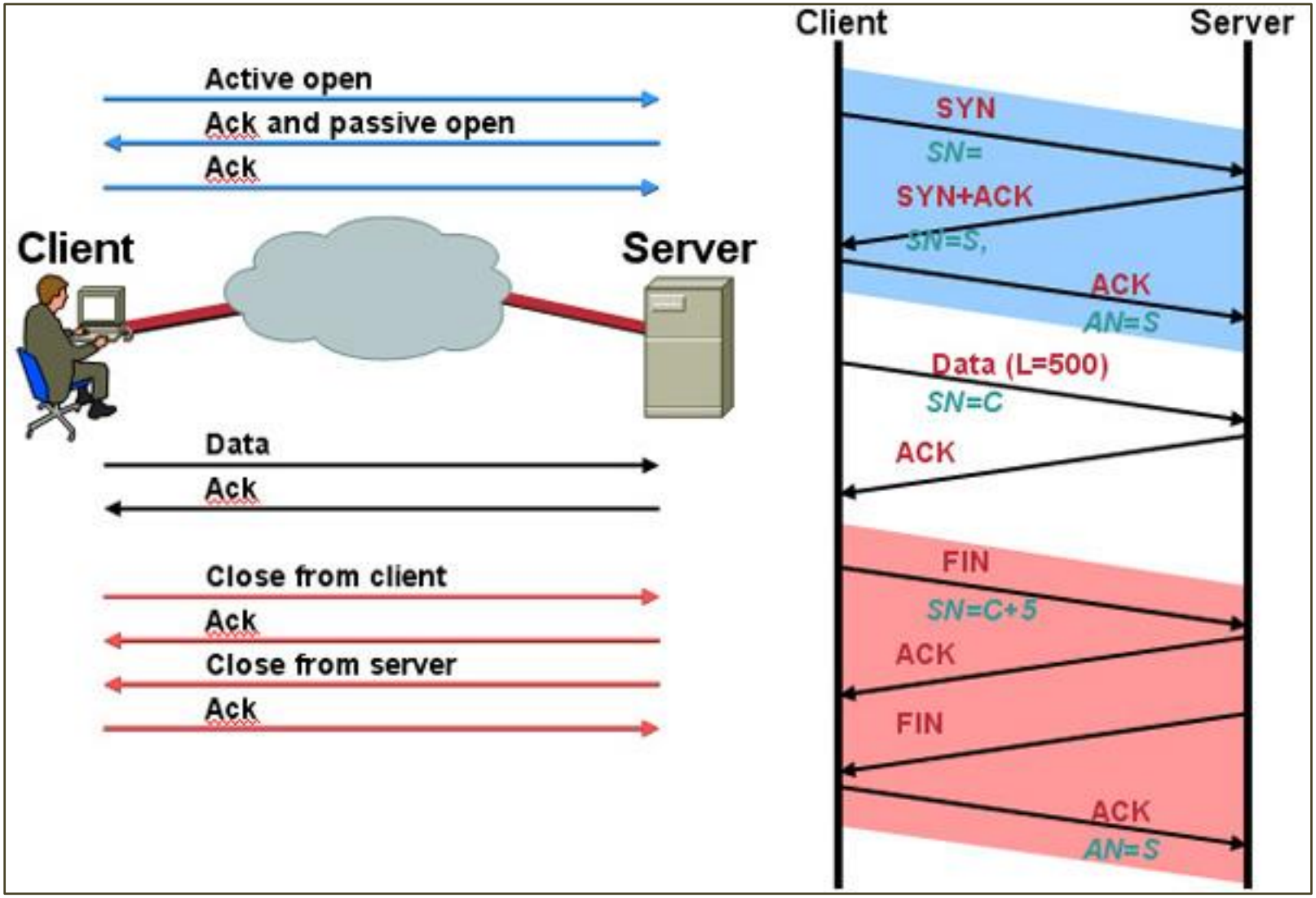
回首頁

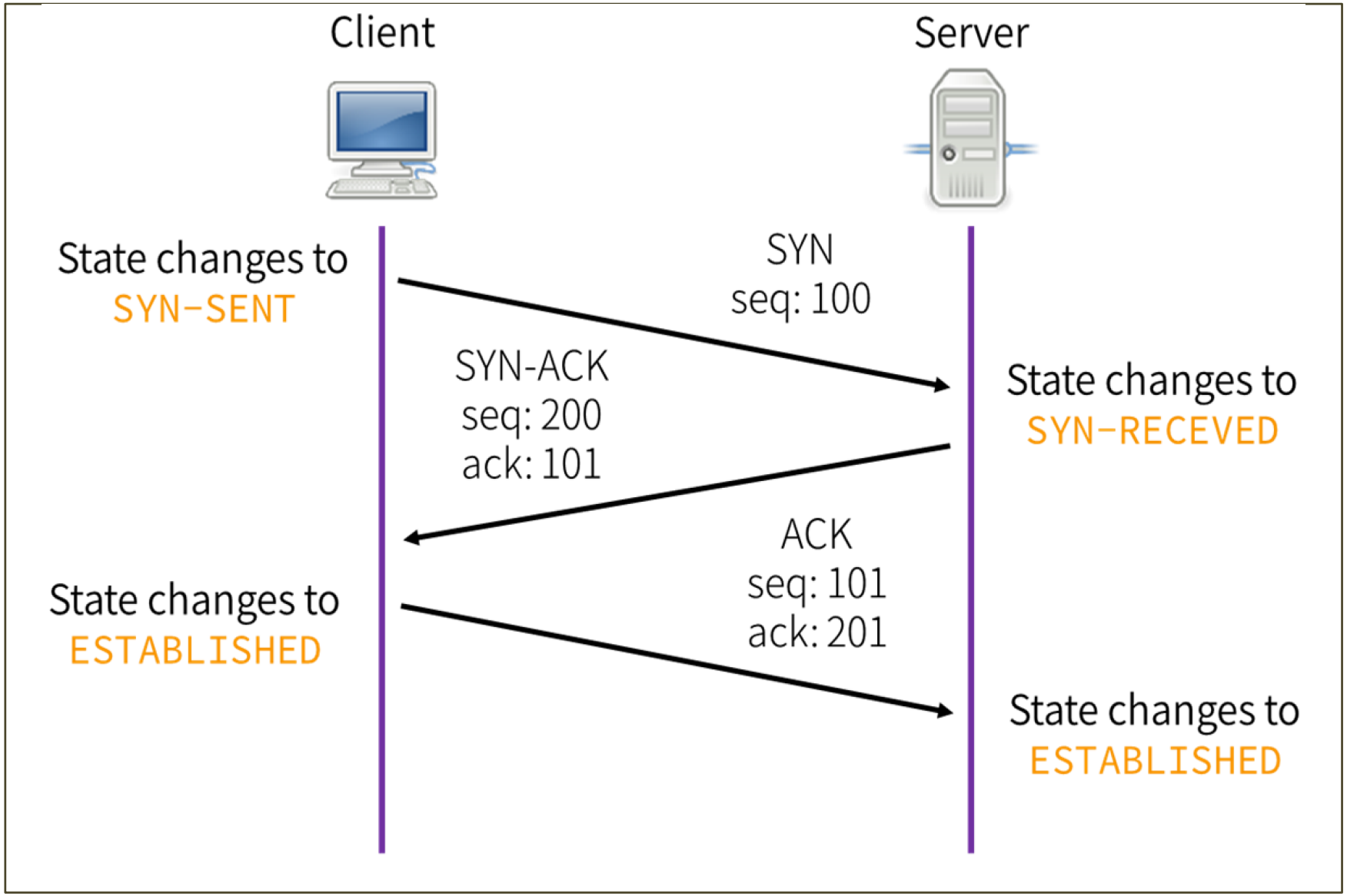
主動式作業系統的辨識原理

- FIN 封包探測或 XMAS 封包探測
- Bogus（偽造）封包探測
- TCP initial windows（TCP 初始化視窗）探測
- ICMP TOS（Type Of Service）判別

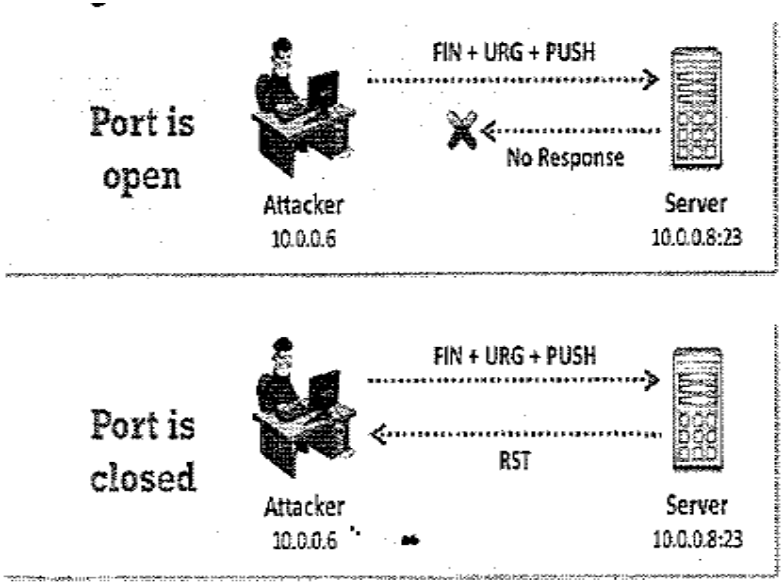
FIN 封包探測或 XMAS 封包探測

- 掃描軟體送出一個 FIN 封包（或任何未設置 ACK 或 SYN 標記位元的資料包）至欲被探測的主機上，在正常的 TCP/IP 規範 (RFC 793) 中，對於此類封包是不予理會的，
- 但在某些的作業系統上（如 windows 系統，cisco...等等）將會回應一個 Reset (RST) 封包，掃描軟體即可利用此種特徵來辯識出作業系統。
- 由於此種封包在封包表頭上的長相就像一顆聖誕樹一樣，所以又稱為 XMAS（聖誕節）的封包探測。


Active open
Ack and passive open
Ack
Client
Server
Data
Ack
Close from client
Ack
Close from server
Ack
Client
Server
SYN
SN=
SYN+ACK
SN=S,
ACK
AN=S
Data (L=500)
SN=C
ACK
FIN
SN=C+5
ACK
FIN
ACK
AN=S


Client
Server
State changes to
SYN-SENT
SYN
seq: 100
SYN-ACK
seq: 200
ack: 101
State changes to
SYN-RECEVED
ACK
seq: 101
ack: 201
State changes to
ESTABLISHED
State changes to
ESTABLISHED

Xmas scan

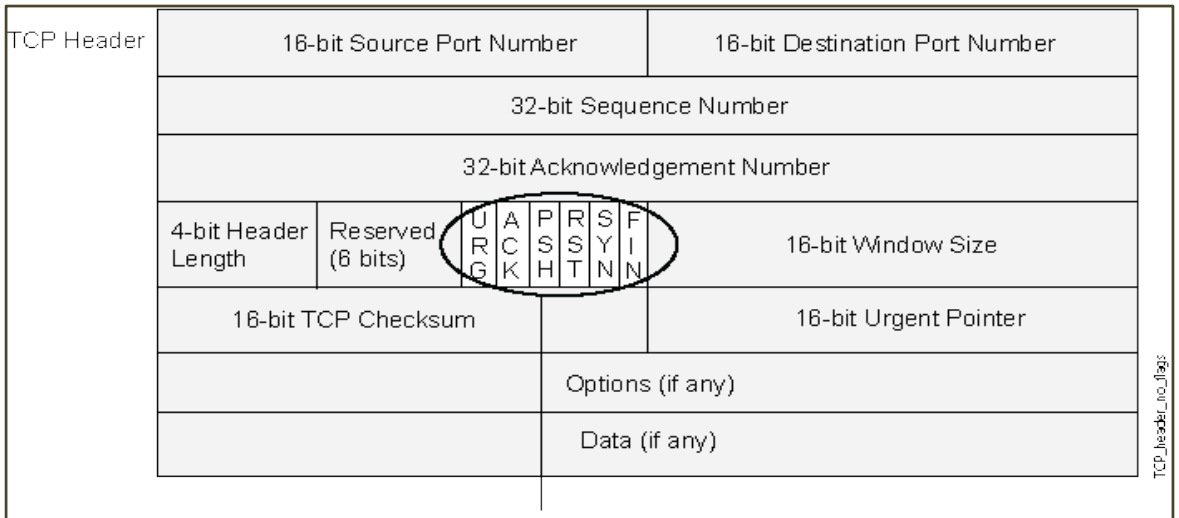

Port is open
FIN + URG + PUSH
No Response
Attacker
10.0.0.6
Server
10.0.0.8:23
Port is closed
FIN + URG + PUSH
RST
Attacker
10.0.0.6
Server
10.0.0.8:23

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 54497 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
2	0.000070000	172.16.20.10	172.16.30.132	TCP	54	54497 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	0.000084000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 24248 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
4	0.000104000	172.16.20.10	172.16.30.132	TCP	54	24248 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	0.000737000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 53651 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
6	0.000802000	172.16.20.10	172.16.30.132	TCP	54	53651 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.000823000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 20315 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
8	0.000846000	172.16.20.10	172.16.30.132	TCP	54	20315 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	0.001208000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 11545 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
10	0.001269000	172.16.20.10	172.16.30.132	TCP	54	11545 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	0.001730000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 37815 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
12	0.001789000	172.16.20.10	172.16.30.132	TCP	54	37815 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	0.002126000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 25802 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
14	0.002185000	172.16.20.10	172.16.30.132	TCP	54	25802 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	0.002642000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > nxlmd [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
16	0.002702000	172.16.20.10	172.16.30.132	TCP	54	nxlmd > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	0.003047000	172.16.30.132	172.16.20.10	TCP	60	ftp-data > 9739 [FIN, PSH, URG] Seq=1 Win=8192 Urg=0 Len=0
18	0.003106000	172.16.20.10	172.16.30.132	TCP	54	9739 > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

```
▷ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▷ Ethernet II, Src: Vmware_79:96:d8 (00:0c:29:79:96:d8), Dst: [redacted]
▷ Internet Protocol Version 4, Src: 172.16.30.132 (172.16.30.132), Dst: 172.16.20.10 (172.16.20.10)
▽ Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 54497 (54497), Seq: 1, Len: 0
    Source port: ftp-data (20)
    Destination port: 54497 (54497)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Header length: 20 bytes
  ▽ 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
    Window size value: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: -1 (unknown)]
  ▷ Checksum: 0x3017 [validation disabled]
    Urgent pointer: 0
```

Bogus（偽造）封包探測

- 掃描軟體在 SYN 封包的表頭上 (header) 設置一個未定義的 TCP 旗標 (flag) 值，在正常的 TCP/IP 規範規定對於此類封包是不予理會，
- 但在一些作業系統上在遇到此類封包 (SYN+Bogus) 時，將會回應 Reset 封包來重置連線。
- 掃描軟體即可利用此種特徵來辯識出作業系統。




```
▷ Frame 1 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: AsustekC_b3:01:84 (00:1d:60:b3:01:84), Dst: Actionte_2f:47:87 (00
▷ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 174.143.213.184 (174.143.21
▽ Transmission Control Protocol, Src Port: 54841 (54841), Dst Port: http (80), Seq: 0
    Source port: 54841 (54841)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
  ▽ Flags: 0x02 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgement: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
    ▷ .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 5840
  ▷ Checksum: 0x85f0 [validation disabled]
  ▷ Options: (20 bytes)
```

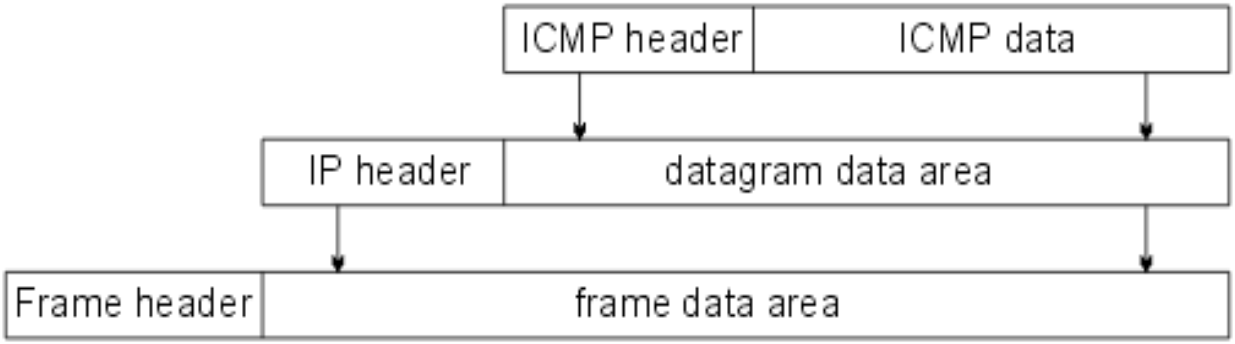
```
0000  00 26 62 2f 47 87 00 1d  60 b3 01 84 08 00 45 00   .&b/G... `.....E.
0010  00 3c 47 65 40 00 40 06  ad 64 c0 a8 01 02 ae 8f   .<Ge@.@. .d......
0020  d5 b8 d6 39 00 50 f6 1c  6c be 00 00 00 00 a0 02   ...9.P.. l.......
0030  16 d0 85 f0 00 00 02 04  05 b4 04 02 08 0a 00 0d   ........ ........
0040  2b db 00 00 00 00 01 03  03 07                     +........ ..
```

以 TCP initial window（TCP 初始化視窗）探測

- TCP initial window 是一種控制網路流量的機制。在傳送的過程中，如果已傳送了視窗 (window) 大小的封包，即表示需接收到對方的 ACK 回應後，才能繼續傳送。
- 但在某些作業軟體上，視窗 (window) 的長度是固定的，所以掃描軟體可利用此種特徵來辨識出作業系統。

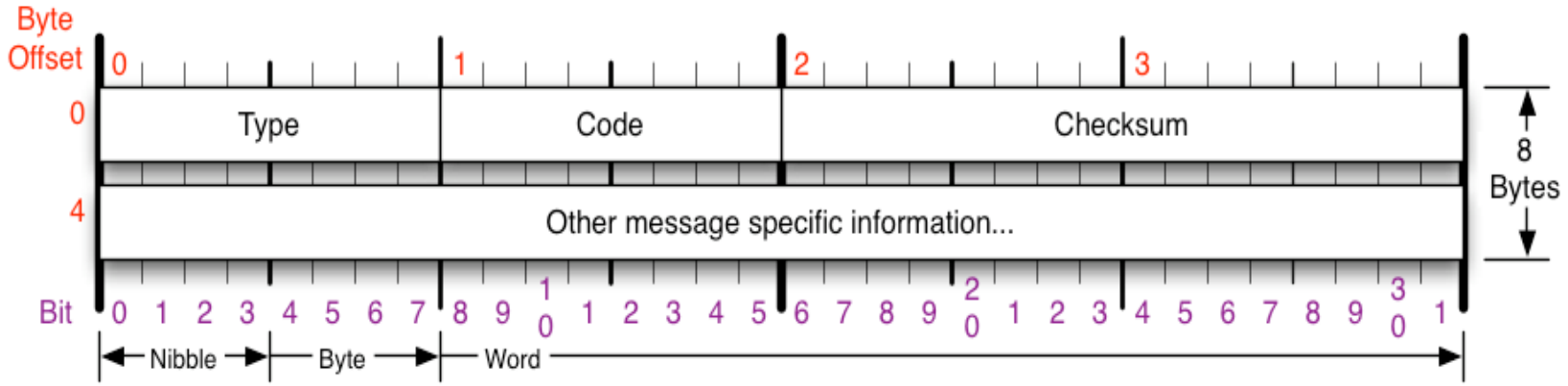
Values for the Operating Systems

Operating System	Time To Live	TCP Window Size
Linux (Kernel 2.4 and 2.6)	64	5840
Google Linux	64	5720
FreeBSD	64	65535
OpenBSD	64	16384
Windows 95	32	8192
Windows 2000	128	16384
Windows XP	128	65535
Windows 98, Vista and 7 (Server 2008)	128	8192
iOS 12.4 (Cisco Routers)	255	4128
Solaris 7	255	8760
AIX 4.3	64	16384



ICMP encapsulation

ICMP Header



ICMP Message Types

Type	Code/Name
0	Echo Reply
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	4 Fragmentation required, and DF set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	9 Network Administratively Prohibited
	10 Host Administratively Prohibited
	11 Network Unreachable for TOS
3	Destination Unreachable (continued)
	12 Host Unreachable for TOS
	13 Communication Administratively Prohibited
4	Source Quench
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the TOS & Network
	3 Redirect Datagram for the TOS & Host
8	Echo
9	Router Advertisement
10	Router Selection
11	Time Exceded
	0 TTL Exceeded
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer Problem
	1 Missing a Required Operand
	2 Bad Length
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
30	Traceroute

Checksum

Checksum of ICMP header

RFC 792

Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification.

ICMP Type	ICMP Code	Description
0	0	Echo Reply (used by ping)
3	0	Destination Network Unreachable
3	1	Destination Host Unreachable
3	3	Destination Port Unreachable
8	0	Echo Request (used by ping)
11	0	TTL Expired (used by traceroute)

ICMP TOS (Type Of Service) 判別

- ICMP 全名為 Internet Control Message Protocol（網際網路訊息控制協定），基本上是一個錯誤偵測與回報的機制，通常是用來檢驗網路的連線狀態與連線的正確性。
- ICMP 封包除了可用來確認主機的狀態外，也可利用探測作業系統的種類，
- 探測程式可利用 ping 程式送出一個 icmp echo 的請求至要探測的主機上，再由對方主機回的 echo reply 封包中的 TTL (Time To Live) 的欄位值（不同作業系統對於 TTL 欄位值的設定都不一樣）來初步判斷主機的作業系統。

- 可利用 ping 指令來觀察 linux 系統及 windows 系統回傳封包的 TTL 欄位，如下所示：
- ping 來測試其回應值，即可發現兩種作業系統所回應封包中的 TTL 值明顯不同。
- 掃描軟體即可利用此種特徵來初步區分作業系統的種類，如下圖為 linux 系統及 windows 系統的測試畫面：

```
[root@spampc ~]# ping -c 3 140.117.12.14
PING 140.117.12.14 (140.117.12.14) 56(84) bytes of data.
64 bytes from 140.117.12.14: icmp_seq=1 ttl=126 time=0.489 ms
64 bytes from 140.117.12.14: icmp_seq=2 ttl=126 time=0.253 ms
64 bytes from 140.117.12.14: icmp_seq=3 ttl=126 time=0.257 ms
```

WINDOWS 系統

```
[root@spampc ~]# ping -c 3 140.117.100.168
PING 140.117.100.168 (140.117.100.168) 56(84) bytes of data.
64 bytes from 140.117.100.168: icmp_seq=1 ttl=64 time=0.374 ms
64 bytes from 140.117.100.168: icmp_seq=2 ttl=64 time=0.148 ms
64 bytes from 140.117.100.168: icmp_seq=3 ttl=64 time=0.151 ms
```

LINUX系統

由上可明顯的看出不同的作業系統會設定不同的 TTL 值，探測軟體即可利用此種特性來初步判別作業系統的種類。

另外一種探測方式是利用管理者不當的設定系統組態而造成作業系統資訊外洩，讀者可利用如 telnet 即可能獲知作業系統的資訊。如下圖即可得知該主機的作業系統：

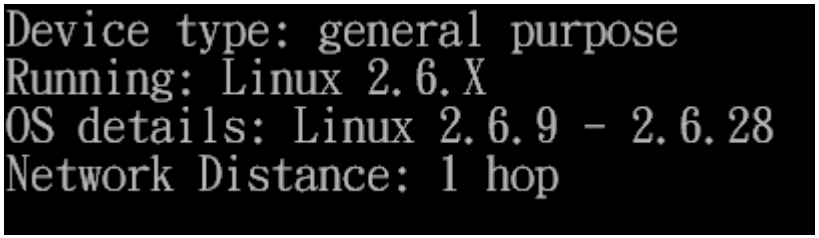
```
playground~> telnet [redacted]
Trying 163.143.103.12...
Connected to hpux.u-aizu.ac.jp.
Escape character is '^]'.

HP-UX hpux B.10.01 A 9000/715 (ttyp2)

login:
```

- dnf 是一個自動安裝工具，它可以幫助RPM系統安裝，移除，升級軟體套件（應用程式，函式庫等）。(fedora 33 later)
- sudo dnf install nmap (如果不行，直接下指令nmap）

另外讀者也可利用現成的工具來探測主機的作業系統，如使用 nmap – O 127.0.0.1 來偵測主機的作業系統，如下圖所示：

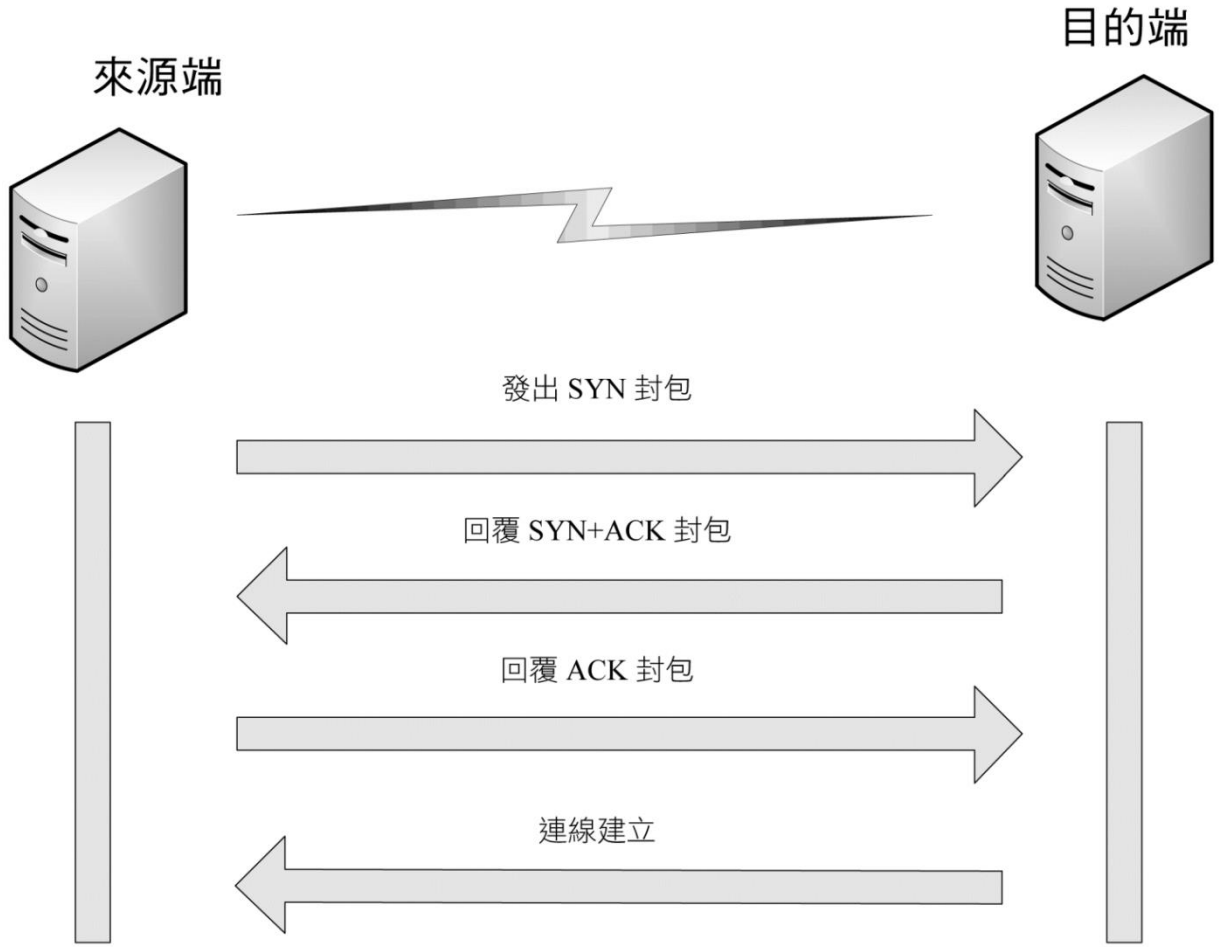


埠掃描 (port scan) 原理

- 在確定主機的作業系統後，接下來的步驟即為確認該主機上運作的服務資訊以便擬定攻擊策略。
- 主機服務的偵測主要是利用被探測主機對於探測封包的回應情況，來判別被探測主機的連接埠 (Port) 是否為開啟的情況。
- 所使用的探測技術，如下所述：

探測主機的通訊埠（Port）

- 全連接技術



- 一旦掃描軟體被探測主機順利完成三向交握 (Three Handshake) 並與目標主機完成連線，即可代表該探測埠處於開啟的狀態。
- 反之即代表探測埠處於關閉的狀態。此種掃描方式即稱為全連接掃描。
- 全連接掃描由於已完成正常的連線，所以此種掃描會被防火牆或 IDS（入侵偵測系統）所記錄。

半連接探測技術

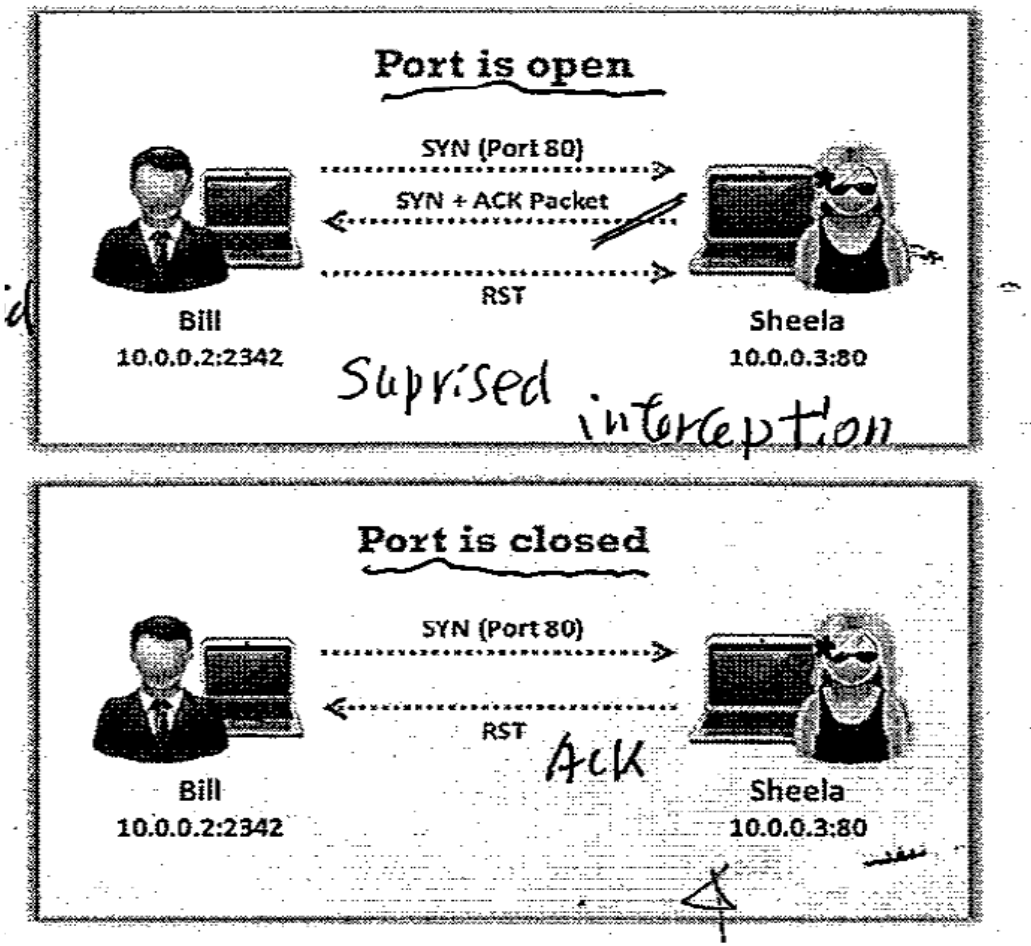
由於使用全連接方式的掃描，很容易即被被防火牆或IDS（入侵偵測系統）所記錄，所以一般的埠探測技術大都不會與被探測主機完成正常的連線以避免被記錄，此種的探測技術稱為半連接探測技術，常用的半連接探測技術，敘述如下：

- SYN 封包掃描
- SYN/ACK 封包掃描技術
- FIN 封包掃描技術

SYN 封包掃描

在上述的三向交握 (Three Handshake) 機制中，當傳送端主機要傳送資訊至接收端主機時，會先發出 SYN 封包到接收端主機，一旦接收端主機接收到後，會回覆 SYN/ACK 封包給傳送端主機，而後傳送端主機將再回覆 ACK 封包至接收端主機，至此連線階段建立，方可開始傳送資訊。

- 而 SYN 封包掃描即是在發出 SYN 封包至對方主機後，如果對方主機有正常的回應 SYN/ACK 封包，即表示連接埠 (Port) 是開啟的情況，並立即發出 RST 封包，切斷該次連線。
- 由於並未完整的完成三向交握 (Three Handshake) 連線。所以防火牆或 IDS（入侵偵測系統）並不會記錄該次的連線。

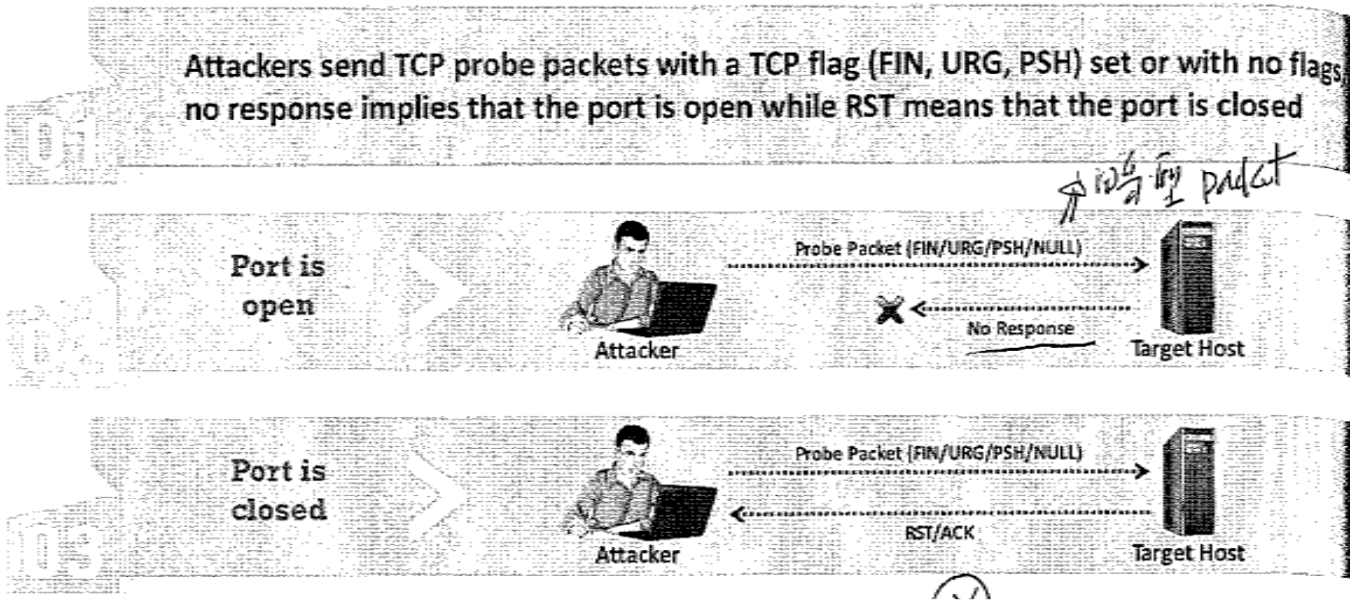

Port is open
SYN (Port 80)
SYN + ACK Packet
RST
Bill
10.0.0.2:2342
Sheela
10.0.0.3:80
Suprised interception
Port is closed
SYN (Port 80)
RST
ACK
Bill
10.0.0.2:2342
Sheela
10.0.0.3:80

SYN/ACK 封包掃描技術

- SYN/ACK 掃描是利用繞過 Three Handshake 第一步的 SYN 封包傳送，而直接利用傳送 SYN/ACK 封包至目的主機上的埠口。
- 由於 TCP 協定具有連接性的性質，當目的埠口接收到此封包後，如果它是開啟的狀態，即會知道此封包並不是合法的封包（因為沒有相對應 SYN 封包），而直接丟棄 (drop)。但如果目的埠口是關閉的狀態，則會回傳 RST 封包（直接重置該連線），
- SYN/ACK 掃描即可利用此種特性來確認連接埠 (Port) 是否在開啟的狀態。

FIN 封包掃描技術

- FIN 掃描是利用繞過三向交握 (Three Handshake) 第一步的 SYN 封包傳送，而直接利用傳送 FIN 封包至目的主機上的埠口。
- 當目的埠口接收到此封包後，如果它是開啟的狀態，即會知道此封包並不是合法的封包（因為沒有相對應 SYN 封包），而直接丟棄 (drop)。但如果目的埠口是關閉的狀態，則會回傳 RST 封包（重置該連線）。
- FIN 掃描即可利用此種特性來確認埠口 (Port) 是否在開啟的狀態，


Attackers send TCP probe packets with a TCP flag (FIN, URG, PSH) set or with no flags
no response implies that the port is open while RST means that the port is closed
Port is open
Probe Packet (FIN/URG/PSH/NULL)
No Response
Attacker
Target Host
Port is closed
Probe Packet (FIN/URG/PSH/NULL)
RST/ACK
Attacker
Target Host

讀者可利用下列指令來掃描被探測主機所開啟的埠及服務資訊：

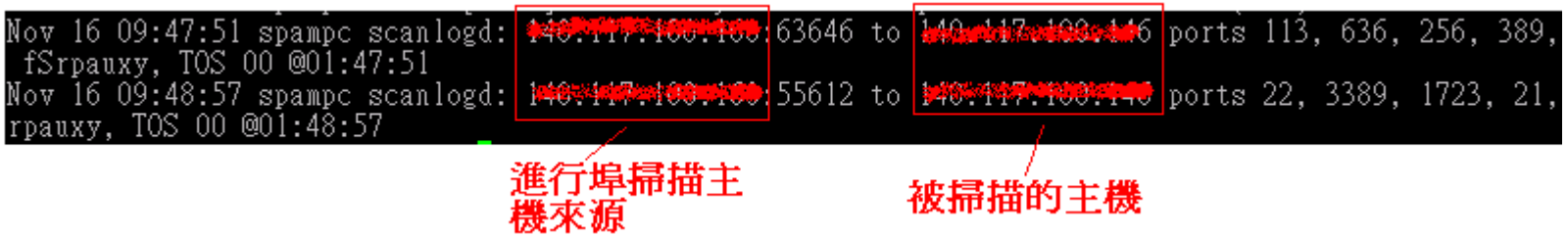
- nmap -sS 127.0.0.1

#利用 syn 掃描來探測被探測主機所開啟埠資訊

```
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
111/tcp  open  rpcbind
443/tcp  open  https
873/tcp  open  rsync
3306/tcp open  mysql
```

2.4 實作步驟

- 利用 scanlogd 埠掃描偵測軟體來建立埠掃描偵測系統，一旦發現有惡意的埠掃描行為，便立即將相關的資訊寫入 /var/log/messages(要另外設定)



- 依下列步驟，安裝 scanlogd 軟體（可至官方網站下載 scanlogd ）。
- Fedora 26以後版本，scanlogd已內建

回首頁

Nmap ("Network Mapper")

- A free and open source (license) utility for network discovery and security auditing
- 操作手冊

http://nmap.org/book/zenmap.html

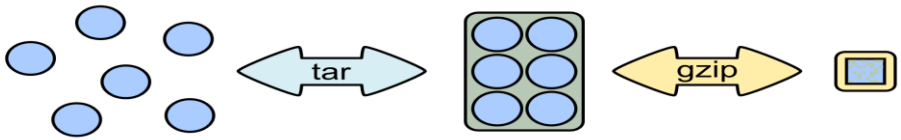
指令名稱	指令說明
adduser scanlogd	新增 scanlogd 使用者來啟動 scanlogd 程式
dnf install libpcap	安裝 libpcap 程式庫 dnf install gcc
tar xvzf scanlogd-2.2.6.tar.gz	解壓縮原始碼
cd scanlogd-2.2.6	至原始碼目錄
make linux	編譯原始碼，編譯成功後在該目錄下會有名稱為 scanlogd 的執行檔
cp scanlogd /usr/bin	將 scanlogd 複製至 /usr/bin/ 目錄下
scanlogd &	執行 scanlogd
ps aux | grep scanlogd	檢查 scanlogd 是否已以常駐程式的形式執行

- Fedora 26以後版本，scanlogd已內建，可以用下列指令檢查

```
ps aux|grep scanlogd
```

- RPM 指的就是 Red Hat Package Manager， 稱之為「包裝檔案管理程式」。
- Libpcap（Packet Capture library），即封包擷取函數庫，其功能是通過網卡抓取網絡乙太網路中的封包。

- Unix和類Unix系統上的壓縮打包工具，可以將多個文件合併為一個文件，打包後的文件名亦為「tar」。
- 最初的設計目的是將文件備份到磁帶上(tape archive)，因而得名tar。
- tar代表未被壓縮的tar文件。已被壓縮的tar文件則追加壓縮文件的延伸檔名，如經過gzip壓縮後的tar文件，延伸檔名為「.tar.gz」。



- -x，--extract，--get 解開tar文件
- -z，--gzip，--gunzip，--ungzip 調用gzip執行壓縮或解壓縮
- -v，--verbose 列出每一步處理涉及的文件的信息，
  - 只用一個「v」時，僅列出文件名，
  - 使用兩個「v」時，列出許可權、所有者、大小、時間、文件名等信息。
- -f，--file [主機名:]文件名 指定要處理的文件名。可以用「-」代表標準輸出或標準輸入。

ps ：將某個時間點的程序運作情況擷取下來

選項與參數：

-a ：顯示其他用戶啟動的所有 process ；

-u ：查看系統中屬於自己的 process ；

-x ：啟動這個行程的用戶和它啟動的時間

- “|” =>將某個指令結果輸出到另一指令
- grep名稱來自於g/re/p（globally search a regular expression and print，以正規表示法進行全域尋找以及列印）
  - 將所有符合先定義樣式的字串，以行為單位列印出來

Regular expression

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	

Fedora 33以後版本，使用下列指令安裝

```
sudo dnf -y install nmap
```