iptables 是 Linux 上進行封包過濾的工具, 不論實作防火牆、NAT、連線管理等各種技術, 都必須使用 iptables。正因爲 iptables 的功能強大, 所以其語法與參數等也比一般指令複雜許多, 常常讓初學者一頭霧水, 形成學習上的障礙。

雖然說學習沒有速成法, 但是只要把握住觀念要領, 還是可以事半功倍, 況且 iptables 的參數看似浩如繁星, 但是常用的其實不多, 所以本章將讓您快速掌握 iptables 的要領, 並且瞭解其常用的語法與參數, 不必苦讀十年, 就能開始利用 iptables 實作進階技術。

本章只解說基本語法與參數, 隨後章節若遇到特殊選項時, 會另外加以說明。此外, 附錄 B 有更詳細的 iptables 說明。

2-1 安檢利器 － iptables

iptables 的功用在於過濾封包, 如果您對於封包過濾的觀念還不清楚, 請試試看在 Linux 主機執行下面指令：

```
[root@free ~]# iptables -I OUTPUT -d www.google.com.tw -j DROP
```

封包過濾屬於系統設定與管理的一部份, 所以必須具備 root 權限才能使用 iptables 處理封包。

然後於此主機上使用瀏覽器連線 "http://www.google.com.tw", 您將發現雖然其它網站仍可連線, 但再也無法連線到 Google 網站了。

上面指令會在 iptables 中建立一個規則, 設定阻擋通往 www.google.com.tw 的連線, 然後 iptables 會過濾所有網路封包, 當遇到符合此規則的封包, 便禁止其通過。

所以 iptables 跟海關、大門警衛等類似, 可以一一過濾進出者, 一旦攔截到符合某個特徵者, 就會立即處理。

2-2 iptables 的語法格式

iptables 最大的功用就是過濾進出主機的封包, 而機場的海關負責過濾進出本國的旅客, 兩者的角色類似, 所以隨後筆者將以海關爲例, 說明 iptables 的語法。

假設您是海關的主管, 如果想要禁止 A 國來的旅客入境, 應該會下達這樣的指令：『檢查所有進來的人, 如果來自 A 國, 一律加以禁止』。這個指令可以拆解成下面三個部分：

1. **檢查的目標**：檢查所有進來的人。
2. **比對的規則**：如果來自 A 國。
3. **處理的動作**：一律加以禁止。

而一個 *iptables* 指令也能如下拆解成同樣的三個部分：

```
iptables -I INPUT -s 192.168.0.139 -j DROP
```

- `-I INPUT`：檢查的目標：檢查所有進入的封包
- `-s 192.168.0.139`：比對的規則：如果來自 192.168.0.139
- `-j DROP`：處理的動作：一律加以禁止

執行上面指令之後, IP 爲 192.168.0.139 的電腦便無法連線至您的主機了。如此拆解之後, 便可以發現 iptables 規則其實並不難瞭解。

一般 *iptables* 指令中,『檢查的目標』與『處理的動作』便如上面例子一樣簡單, 不過『比對的規則』可就千變萬化了。複雜的比對規則, 讓 iptables 的功能因此而強大, 卻也常常讓初學者眼花撩亂, 但是不論是什麼樣的 *iptables* 指令, 只要先將其拆解成前面所說的三個部分, 就能讓複雜度減少一大半了。

後面筆者將分別就此三個部分, 依序說明各部分的觀念與使用方法。

2-3 利用鏈與表格設定檢查目標

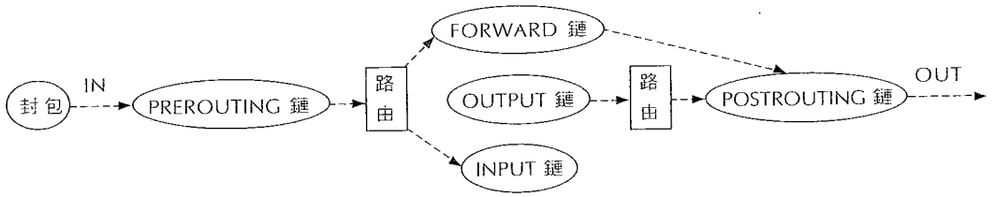
首先將爲您說明『檢查的目標』, 此部分雖然不像『比對的規則』那樣充滿複雜的參數, 但是在觀念上卻很容易讓人混淆。所以請特別注意概念與原理說明, 只要觀念清楚, 就能順利掌握此部分。

鏈的功用

在海關中, 旅客會依照出境、入境與過境等差異, 而需要通過不同的檢查處。同樣的, 網路封包也需要通過不同的檢查處, 而在 iptables 中, 將各個檢查處稱爲『鏈』(chain)。

所以當網路管理者需要處理特定封包時, 只要依照鏈的性質不同, 在適當的鏈設定規則即可。就像海關如果要管制進入的旅客, 就要在入境處進行檢查, 而過濾出國旅客時, 就要在出境處檢查一樣。

封包與鏈的關係示意圖如下：



所有 iptables 預設的鏈, 皆使用大寫字母爲其名稱。

下面說明各個鏈的意義：

- **PREROUTING**：由上圖可以看到, 當封包進入主機時, 首先通過的是 PREROUTING 鏈。"pre" 字首爲『預先』的意思, 而在 TCP/IP 的觀念中, routing 則是一個動詞, 通常譯爲『路由』, 代表封包該往何處傳送的判斷動作。路由的作用有點類似機場的航班時刻表與廣播, 負責指引各航線的旅客應該走哪一個登機門。在進行路由判斷前, 還不確定封包應送往本機或是其他主機, 便會先通過 PREROUTING 鏈。您可以將此鏈想像成機場大門, 旅客到達機場後, 必須先通過大門, 然後才是辦理手續以確定自己應該前往哪一個登機門。

關於路由的詳細說明, 請參考 1-2 節。

- **INPUT**：INPUT 是**封包送入主機的網路程序前**需要通過的鏈, 其作用與入境海關類似, 旅客在進入本國前, 都必須先通過海關的入境處。同樣的, 封包經過路由判斷後, 發現**目的地爲本機**, 也就是想要進入主機、連線主機內網路程序的封包, 都會交由 INPUT 鏈處理。

- **OUTPUT**：與 INPUT 鏈相反, OUTPUT 鏈爲**主機的網路程序送出封包前**需通過的鏈, 類似出境海關, 主機網路程序所有對外送出的封包, 都必須由 OUTPUT 鏈檢查。

- **FORWARD**：FORWARD 鏈負責處理轉送的封包, 所謂轉送, 指的是封包經過路由判斷之後, 發現**其目的地爲其他主機**, 所以封包不會送入本機的任何網路程序, 而會直接從另一個網路介面出去, 此時該封包會由 FORWARD 鏈處理。您可以將 FORWARD 鏈類比爲過境時的檢查點 (雖然現實生活中, 過境轉機大多不需要檢查), 轉機的旅客目的地是其他國家, 在本國機場下了飛機後, 隨即會轉搭另一飛機到其他國家, 不會進入本國的領土。因此不像 INPUT 鏈與 OUTPUT 鏈只單純負責接收與送出封包, FORWARD 鏈會同時有接收與送出封包的處理動作。

- **POSTROUTING**：對外傳送的封包通過 OUTPUT 鏈或 FORWARD 鏈之後, 最後還會再由 POSTROUTING 鏈檢查, 類似機場的登機門, 一旦通過 POSTROUTING 鏈後, 便可以順利傳送至外面。由於到達 POSTROUTING 鏈的封包皆已經過路由判斷確定要送達的目的地, 所以其名稱的字首爲 "post", 表示路由『之後』的意思。

當管理者想要限制特定封包進入Linux 主機時, 例如要阻擋某個 IP 連線到主機的 HTTP 伺服器, 則應該在 INPUT 鏈設定規則；若希望過濾 Linux 主機對外送出的封包時, 例如想要管制主機上某個使用者不得對外連線, 便需要在 OUTPUT 鏈建立規則。

而如果 Linux 主機擔任內部與外部網路之間的閘道, 內外部網路連線的封包並不會送入主機的任何網路程序, 也不會被主機網路程序送出, 此時封包從某個介面進入後只是單純地立刻轉送至另一介面, 所以若想要管理內部與外部網路的連線時, 應使用 FORWARD 鏈。

請務必分清楚 INPUT、OUTPUT 與 FORWARD 鏈的差別, 因爲這三個鏈與防火牆的設定息息相關。此外, PREROUTING 與 POSTROUTING 鏈一般情形下使用率較低, 所以隨後章節如果需要在這兩個鏈設定規則時, 會詳細爲您說明。

封包會依序進入各個鏈進行處理, 如果前面的鏈設定某個封包允許通過, 並不代表該封包就此一路順利, 因爲仍有可能會被後面的鏈阻擋。而封包一旦在某個鏈遇到禁止通過的規則, 便會就地正法, 就算後面的鏈設定允許其通過的規則也是枉然, 因爲封包已經在前面的鏈被阻擋了。

除了預設的鏈以外, iptables 還允許管理者自行建立新的鏈, 本章稍後內容會簡單說明如何自訂新鏈, 而其他章節則會以實際的例子, 解說如何自訂新鏈來達成所需的功能。

表格的功用

在 iptables 發展出來之前, Linux 使用 ipchains 進行封包過濾的工作, 顧名思義, ipchains 的主要單位爲鏈 (chain), 因此前面所提到的鏈, 是在 ipchains 就已經出現的概念。不過到了 iptables 設計時, 則加入了一個新的概念：表格 (table)。就像我們會將不同用途的工具分門別類, 依照功能的不同, iptables 共有以下三個表格：

- **filter**：這個表格可設定封包的過濾規則, 在第 3、4 章設定防火牆時, 主要會用到這個表格。
- **nat**：這個表格主要是用來轉換 IP 位址, 讓內部使用私有 IP 的電腦可以連線網際網路, 第 7 章會詳細說明此表格。
- **mangle**：這個表格可做進階的封包管理, 本書第 4 篇部分章節會爲您介紹此表格的應用。

與鏈名皆使用大寫字母不同, 所有 iptables 表格的名稱皆爲小寫。

當管理者想要設定過濾的規則, 建立防火牆時, 大多數的情況下應該將規則放置於 filter 表格, 而 nat 表格內則主要放置 NAT (Network Address Translation, 網路位址轉換, 第 7 章會說明此技術) 相關的規則, 這樣就不會讓所有規則通通放在一起, 造成使用上的混淆與困擾。

由於這些表格性質上的差異, 所以每個表格內包含以下不同的鏈：

- **filter**：此表格內包含 INPUT、OUTPUT、FORWARD 三個鏈。
- **nat**：此表格內包含 PREROUTING、OUTPUT、POSTROUTING 三個鏈。
- **mangle**：在核心 2.4.17 版之前, 此表格只包含 PREROUTING 與 OUTPUT 兩個鏈, 從 2.4.18 版之後, 包含了所有 iptables 的五個鏈：PREROUTING、INPUT、OUTPUT、FORWARD、POSTROUTING。

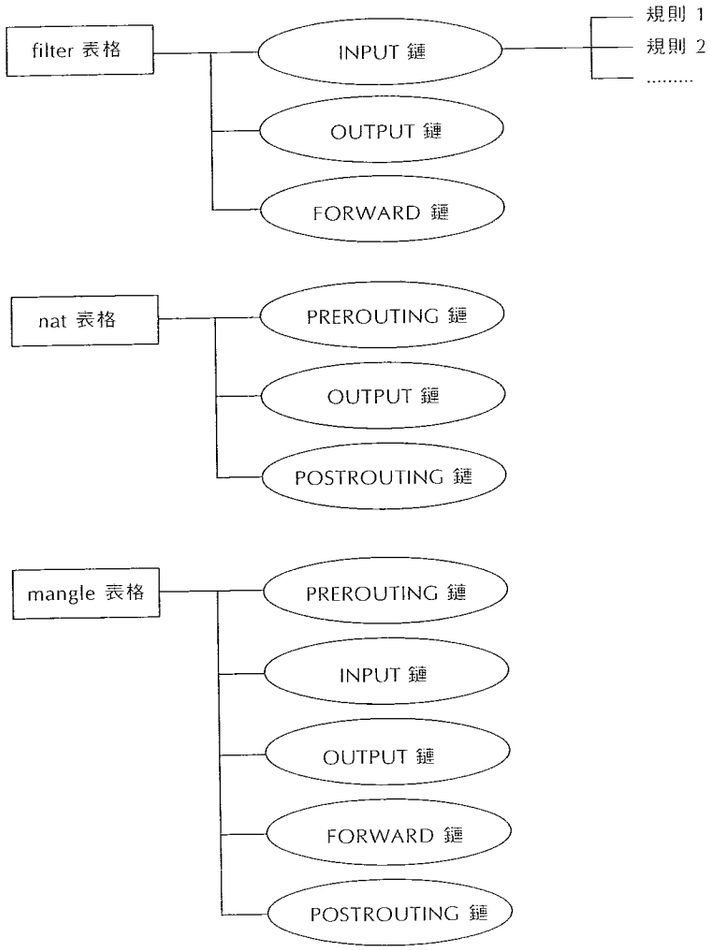
雖然鏈名一樣, 但是 filter 表格與 nat 表格的 OUTPUT 鏈仍然是獨立分開的兩條鏈。也許您會疑惑, 如果在不同表格但相同名稱的鏈, 例如 filter 表格與 mangle 表格的 INPUT 鏈, 設定完全相反的規則時, 要如何判斷誰先誰後呢？

對於各個表格內相同名稱的鏈, 其先後順序爲：mangle → nat → filter。所以若某個封包在 nat 表格的 OUTPUT 鏈設定爲禁止, 但是在 filter 表格的

OUTPUT 鏈卻是允許通過, 則封包在 nat 表格的 OUTPUT 鏈就會先被阻擋。同樣的道理, 如果 filter 表格的 OUTPUT 鏈設定禁止某個連線, 那麼不論在 mangle 與 nat 表格的 OUTPUT 鏈設定什麼規則, 最後到了 filter 表格的 OUTPUT 鏈還是無法通過。

iptables 的架構

綜合前面所述, iptables 的整個架構就如下圖所示：



當封包進入了某個鏈之後, 就會開始與該鏈的規則**依序**進行比對, 一旦某個規則與封包相符合, 就會依照規則上的處理動作, 對封包進行適當的處置, 不會再繼續比對該鏈隨後的規則, 而某個鏈處理完了之後, 如果該封包沒有被阻擋, 則會繼續交由下一個鏈檢查。

設定檢查目標的語法格式

瞭解了鏈與表格的意義與功用之後, 就可以根據檢查目標的性質, 在適當的表格與鏈中建立規則, 例如筆者想要禁止特定 IP 連線至主機, 那麼便應該在 filter 表格的 INPUT 鏈設定比對規則。由此可知, 建立規則時, 應該先指定表格名稱, 然後是鏈名, 最後才是規則。

下表列出本節所介紹的參數與簡要說明：

參數	功能
-t	指定表格名稱
-I	在鏈的最前面插入規則
-A	在鏈的最後面附加規則
-D	在鏈中刪除某個規則
-N	自訂新鏈

指定表格名稱的參數

iptables 指定表格名稱的參數爲 "-t", 所以使用下面指令可以設定使用 nat 表格：

```
[root@free ~]# iptables -t nat ...
```
← 指定使用 nat 表格

若省略此參數, 則 iptables 預設會使用 filter 表格。

鏈名相關參數

下面是 iptables 指定鏈名的參數：

- **-I**：此參數爲 "insert" 的意思, 可以在鏈的最前面加入規則, 當鏈中有好幾條規則時, iptables 在過濾的過程中會依序往下比對, 當遇到符合的規則之後, 就不會再往下比對, 因此如果有需要優先執行的規則, 可用此參數將規則插入鏈的最前面。例如下面指令可以將規則插入 nat 表格 OUTPUT 鏈的最前面：

```
[root@free ~]# iptables -t nat -I OUTPUT ...
```

- **-A**：此參數爲 "append" 的意思, 可以將規則附加到鏈的最後面, 例如下面指令可以將規則附加於 filter 表格的 INPUT 鏈後面：

```
[root@free ~]# iptables -A INPUT ...
```

- **-D**：此參數爲 "delete" 的意思, 可以刪除鏈中的規則, 使用此參數時, 必須完整描述要刪除哪一條規則。例如若您之前執行 *iptables -A FORWARD -p tcp -i eth1 -s 192.168.0.132 -o eth0 -d 10.1.1.2 -j DROP* 將規則附加於 filter 表格的 FORWARD 鏈 (關於規則參數的解釋, 將於稍後的內容說明), 那麼要刪除該規則時, 應執行下面指令：

```
[root@free ~]# iptables -D FORWARD -p tcp -i eth1 -s
  192.168.0.132 -o eth0 -d 10.1.1.2 -j DROP
```

不過除了完整描述以外, 此參數還有更方便的使用法, 就是直接指定規則的編號, 例如 *iptables -t nat -D OUTPUT 3*, 表示要刪除 nat 表格 OUTPUT 鏈的第 3 條規則, 所以如果知道規則所在的次序, 就不需要輸入一長串的指令了。

- **-N**：前面曾經提到 iptables 允許管理者自行建立新的鏈, 若您想要自訂新鏈, 應該使用 "-N" 參數, 例如下面指令會於 nat 表格中, 新增一個名爲 mychain 的鏈：

```
[root@free ~]# iptables -t nat -N mychain
```

2-4 封包比對的規則

在瞭解應如何針對『檢查目標』的性質, 選擇適當的表格與鏈之後, 本節將說明建立『比對的規則』時, 應該使用的參數。此部分的觀念很簡單, 只是參數較多, 所以顯得繁雜, 您可以先略讀, 大致瞭解 iptables 有哪些比對規則, 未來眞正使用時, 再回來查詢即可。

以下爲建立比對規則時, 常用的參數：

參數	功能
-i	指定封包進入的網路介面
-o	指定封包出去的網路介面
-p	指定封包的類型 (TCP、UDP、ICMP)
-s	指定封包的來源 IP 位址
-d	指定封包的目的 IP 位址
--sport	指定封包來源端的通訊埠號
--dport	指定封包目的端的通訊埠號

- **-i**：此參數爲 "in-interface" 的意思, 用以指定封包進入的網路介面, 如 eth0、ppp0。下面指令設定若封包進入的介面爲 eth0, 則加以處理：

```
[root@free ~]# iptables -A INPUT -i eth0 ...
```

在 filter 表格的 INPUT 鏈後面附加此規則

如果封包進入的網路介面爲 eth0

- **-o**：此參數爲 "out-interface" 的意思, 用以指定封包出去的網路介面。下面指令設定若封包出去的介面爲 eth1, 則加以處理：

```
[root@free ~]# iptables -A FORWARD -i eth0 -o eth1...
```

如果封包進入的網路介面爲 eth0

而且出去的介面爲 eth1

- **-p**：此參數爲 "protocol" 的意思, 用以指定封包所屬的類型, 如 tcp、udp 或 icmp。下面指令設定若爲 TCP 封包, 則加以處理：

```
[root@free ~]# iptables -I OUTPUT -o ppp0 -p tcp...
```

在 filter 表格的 OUTPUT 鏈最前面插入此規則

若封包出去的網路介面爲 ppp0

而且如果封包的類型爲 TCP 協定

- **-s**：此參數爲 "source" 的意思, 用以指定封包的來源 IP 位址, 可以是主機名稱或 IP 位址。下面指令設定若封包的來源 IP 爲 192.168.0.137, 則加以處理：

```
[root@free ~]# iptables -t nat -A PREROUTING -s 192.168.0.137...
```

在 nat 表格的 PREROUTING 鏈後面附加此規則

如果封包的來源 IP 爲192.168.0.137

- **-d**：此參數爲 "destination" 的意思, 用以指定封包的目的 IP 位址。下面指令設定若封包的目的 IP 爲 192.168.0.139, 則加以處理：

```
[root@free ~]# iptables -t filter -A OUTPUT -d 192.168.0.139...
```

在 filter 表格的 OUTPUT 鏈後面附加此規則

如果封包的目的 IP 爲192.168.0.139

- **--sport**：此參數爲 "source-port" 的意思, 用以指定 TCP 或 UDP 封包的來源端埠號, 如 80、22, 此參數必須搭配並放置於 "-p" 參數之後。下面指令設定若封包類型爲 UDP, 而且來源端的埠號爲 53, 則加以處理：

```
[root@free ~]# iptables -I FORWARD -p udp --sport 53...
```

在 filter 表格的 FORWARD 鏈最前面插入此規則

若爲 UDP 封包

如果封包來源端的埠號爲 53

- **--dport**：此參數爲 "destination-port" 的意思, 用以指定 TCP 或 UDP 封包的目的端埠號, 此參數必須搭配並放置於 "-p" 參數之後。下面指令設定若 TCP 封包的來源 IP 爲 10.0.0.1, 而且目的端的埠號爲 80, 則加以處理：

```
[root@free ~]# iptables -A INPUT -p tcp -s 10.0.0.1 --dport 80...
```

若爲 TCP 封包

來源 IP 爲 10.0.0.1

且目的端埠號爲 80

上面所述是最常見的規則設定參數, 至於其他使用頻率較低的參數, 請參見附錄 B 的說明。

使用 ！ 符號設定相反規則

設定規則時, 還可以使用 "!" 符號, 表示『相反』的意思, 例如下面指令設定若 UDP 封包的目的埠號爲 53, 但是目的 IP 不爲 10.0.0.2, 則加以處理：

```
[root@free ~]# iptables -I OUTPUT -p udp -d ! 10.0.0.2
--dport 53...
```

2-5 處理封包的動作

前面解說了『檢查目標』與『比對規則』相關的觀念和參數, 再來則要說明如何設定『處理動作』了。

下表爲 iptables 設定處理動作的參數：

參數	功能
-j 處理方式	設定規則的處理方式
-P 處理方式	設定鏈的預設處理原則

設定規則的處理動作

設定規則處理動作時, 皆使用參數 "-j", 這是 "jump" 的意思, 表示符合規則的封包應該『跳』到哪裡進行處理, 類似海關攔截到通緝犯時, 會移交給警察單位一樣。

iptables 常見的處理動作如下：

參數	功能
DROP	丟棄該封包, 且不回應任何錯誤訊息
REJECT	丟棄該封包, 但回應錯誤訊息
ACCEPT	允許封包通過

- **DROP**：表示丟棄該封包, 且不會回應任何錯誤訊息。下面指令設定當連線通過 nat 表格的 PREROUTING 鏈時, 若發現封包的來源 IP 爲 192.168.0.137, 則丟棄該封包, 而且不回應錯誤訊息：

```
[root@free ~]# iptables -t nat -A PREROUTING -s 192.168.0.137
-j DROP
```

- **REJECT**：丟棄該封包且回應錯誤訊息, 此參數與 "DROP" 的差別在於是否回應錯誤訊息, 若使用 "REJECT" 參數回應錯誤訊息, 則對方會發現連線已被阻擋, 而 "DROP" 參數不會讓對方知道連線已被禁止, 所以對方的程式會出現連線逾時 (timeout) 的訊息。
- **ACCEPT**：允許該封包通過。
- **自訂的鏈名**：這裡可以填入自訂的鏈名, 表示符合規則的封包, 應該跳到此自訂的鏈, 由該鏈繼續處理。

設定鏈的預設處理原則

前面提到當封包會依序與鏈中的規則進行比對, 發現符合某個規則後, 就會依照規則上的處理動作, 進行適當的處置。不過如果封包與該鏈所有規則都不相符的時候, iptables 會如何處理呢？

iptables 的每個鏈都有一個處理原則 (也可稱爲政策, policy), 所以當上面情形發生時, 就會依照此原則處理封包。所有處理原則預設的動作皆爲 ACCEPT, 也就是允許封包通過, 您可以使用 "-P" 參數修改處理原則。例如下面指令設定 filter 表格 INPUT 鏈的處理原則爲 DROP：

```
[root@free ~]# iptables -t filter -P INPUT DROP
```

由於 iptables 預設的表格爲 filter, 所以上面指令也可以改成 *iptables -P INPUT DROP*。

請注意, "-j" 參數爲『規則』處理動作的參數, 因此設定鏈的處理原則時, DROP 前面不應該加上 "-j"。至於 "-P" 參數可用的處理動作請參考前面 "-j" 參數的說明。不過, "-j" 參數可以設定封包跳到自訂鏈進行處理, 但是 "-P" 參數不得設定使用自訂的鏈。

一般建立防火牆的時候, 會將所有相關鏈的處理原則改爲 DROP, 預設丟棄所有封包, 然後設定適當放行規則只允許特定封包, 以提高安全性。關於防火牆預設原則的詳細說明, 請參閱 3-1 節。

2-6 iptables 指令的其他常用參數

前面所提到的 iptables 參數, 主要使用於建立規則的時候, 以下將爲您說明 iptables 指令的其他參數：

參數	功能
-L	列出表格中所有規則
-n	不要將 IP 反查爲網域名稱
-F	清除表格中的規則
-X	移除自訂的鏈

- **-L**：列出表格的所有規則, 例如下面指令可列出 nat 表格所有鏈的全部規則：

```
[root@free ~]# iptables -t nat -L
Chain OUTPUT (policy ACCEPT)
target  prot opt source          destination
DROP    tcp  --  anywhere        anywhere          tcp dpt:ftp
DROP    tcp  --  anywhere        ms1.hinet.net     tcp dpt:smtp

Chain POSTROUTING (policy ACCEPT)
target  prot opt source          destination

Chain PREROUTING (policy ACCEPT)
target  prot opt source          destination
```

處理動作 / 封包類型 / 來源位址 / 目的位址 / 其他參數, 例如此處的 dpt 爲 "--dport" 的縮寫

nat 表格的 OUTPUT 鏈總共有 2 個規則

nat 表格的 PREROUTING 與 POSTROUTING 目前沒有規則

"-L" 參數後面也可指定鏈名, 如此只會列出該鏈的規則, 例如 *iptables -t nat -L OUTPUT* 將只列出 nat 表格 OUTPUT 鏈的所有規則。

- **-N**：iptables 列出規則時, 會將 IP 反查爲網域名稱, 如果希望 iptables 只顯示 IP 位址, 可以如下加上 "-n" 參數：

```
[root@free ~]# iptables -t filter -L INPUT -n
Chain INPUT (policy ACCEPT)
target   prot opt source          destination
DROP     tcp  --  168.95.4.10     0.0.0.0/0          tcp dpt:80
                  (直接顯示 IP 位址)
DROP     udp  --  0.0.0.0/0       0.0.0.0/0          udp dpt:53
```

因為 iptables 預設會使用 filter 表格, 所以上面指令也可以改成 *iptables -L INPUT -n*。

- **-F**：清除表格中的規則, 若沒指定鏈名, 預設將清除該表格中所有鏈的全部規則。如 *iptables -t nat -F* 指令, 會清空 nat 表格的所有規則。而 *iptables -F INPUT* 則只會清除 filter 表格中 INPUT 鏈的規則。

- **-X**：此參數可以移除自訂的鏈, 但該鏈必須為空。下面為您舉例說明：

```
[root@free ~]# iptables -X badip  ←移除 filter 表格中自訂的 badip 鏈
iptables: Directory not empty  ←若鏈中有規則存在, 會出現此錯誤訊息
[root@free ~]# iptables -F badip  ←清空 filter 表格中 badip 鏈的規則
[root@free ~]# iptables -X badip  ←此時移除就不會發生錯誤了
[root@free ~]# iptables -X  ←若未指定鏈名, 則會移除 filter 表格中所有自訂的空鏈
```

2-7 iptables 規則的儲存與還原方法

前面在指令列下達 *iptables* 指令所設定的規則, 在主機重新開機後, 所有規則就會消失, 必須重新設定才行。為了避免重新輸入的麻煩, 您可以使用 *iptables-save* 指令, 於重開機前將目前 iptables 的所有規則儲存於檔案中：


```
[root@free ~]# iptables-save > /etc/iptables.save
```

(*iptables-save* 指令能夠顯示目前 iptables 的所有規則；使用輸出重導的功能, 將規則儲存於 /etc/iptables.save)

```
[root@free ~]# cat /etc/iptables.save  ←顯示 /etc/iptables.save 檔案的內容
# Generated by iptables-save v1.3.0 on Thu Mar 31 13:32:21 2005
*nat
:OUTPUT ACCEPT [33:2555]
:POSTROUTING ACCEPT [17:1415]
:PREROUTING ACCEPT [6901:898911]
-A OUTPUT -d 168.95.4.10 -p tcp -m tcp --dport 25 -j DROP
...
```

重開機後則可以如下使用 *iptables-restore* 指令, 將儲存於 /etc/iptables.save 檔案的設定還原：

```
[root@free ~]# iptables-restore < /etc/iptables.save
```

若希望每次開機時, 都自動建立 iptables 規則, 則必須將上述 *iptables-restore < /etc/iptables.save* 指令寫入 /etc/rc.d/rc.local 檔, 便能自動將已存檔的規則還原。當然, 您也可以一一將所有設定規則的 *iptables* 指令寫入 /etc/rc.d/rc.local 檔, 如此開機時, 也會自動重新建立規則。