沒有固定 port 應用程式的偵測與過濾: L7-filter classifier

張朝江 林盈達
國立交通大學資訊科學系
300 新竹市大學路 1001 號
Tel: 03-5712121 ext. 56667
E-MAIL：{tjchang,ydlin}@cis.nctu.edu.tw

摘要

隨著網路的普及及使用者的增加，網路安全的議題顯得愈來愈重要。面對安全維護的問題，首先必須想到就是防火牆的使用。就 Linux 作業系統而言，最著名的防火牆軟體套件便是 IPTables，其內建 NAT、簡單的規則設定、以及可擴充性的掛載其他模組功能 (match modules)，使其成為 Linux 內建的防火牆套件。然而在現今網路的應用軟體中，大量的 dynamic-port 應用軟體被使用，因此使用傳統的 OSI Layer 4 port analysis 已經無法達到準確的封包辨認，必須將封包的辨認層次提升到 OSI Layer 7 application layer 才能精確的辨認封包。本篇將介紹網路套件 L7-filter classifier，此套件是掛在 IPTables Explicit matches 上，藉由使用此套件我們將可以看到封包應用層的資訊 ， 以解決使用 port analysis 無法準確辨別封包的問題。

關鍵字：IPTables，dynamic-port，L7-filter

1. 簡介

在網路的應用不斷增加下，傳統 well-known port 已漸漸顯得不敷使用，使得使用 static port 的網路軟體已經不適合現狀，而發展出 dynamic-port 的網路應用。然而使用 dynamic-port 網路對於傳統的封包辨識上產生難處。在傳統封包辨識主要是依照 OSI Layer 4 port analysis 來辨認，一個應用固定對應一個 port，然而 dynamic-port 的網路技術，再加上近年來許多應用都透過 web 的 port 80 進行，如 Web mail(POP3/HTTP)，使得必須將封包辨認的層次提升到 OSI

Layer 7 application layer 直接去看 message 內容。就 Linux 作業系統而言，傳統使用 IPTables 可以達到封包辨識而加以處理的功能，已無法使用在 dynamic-port 軟體上。在本文中，我們將介紹一套可掛載在 IPTables extended packet matching modules 上的 L7-filter classifier 套件[1]，藉由此套件，我們可以增加 IPTables 的功能，使得封包的辨識層次提升到應用層。而哪些網路應用封包需要使用此套件來做辨識功能呢？主要是用在使用不可預期 port 的網路軟體(例如 peer-to-peer filesharing)，或者是使用非標準 port 的軟體(例如 HTTP on port 1111、在 port 80 進行 Webmail)。

因為 L7-filter 是一個掛在 IPTables 架構下的套件，因此若想對 L7-filter 處理封包方式有深入的了解，必須先知道 IPTables 其處理封包的方式。 所以我一開始會先針對 IPTables 做介紹，讓讀者清楚 IPTables 運作方式後，再描述 L7-filter 的使用方法。

2. 封包處理流程

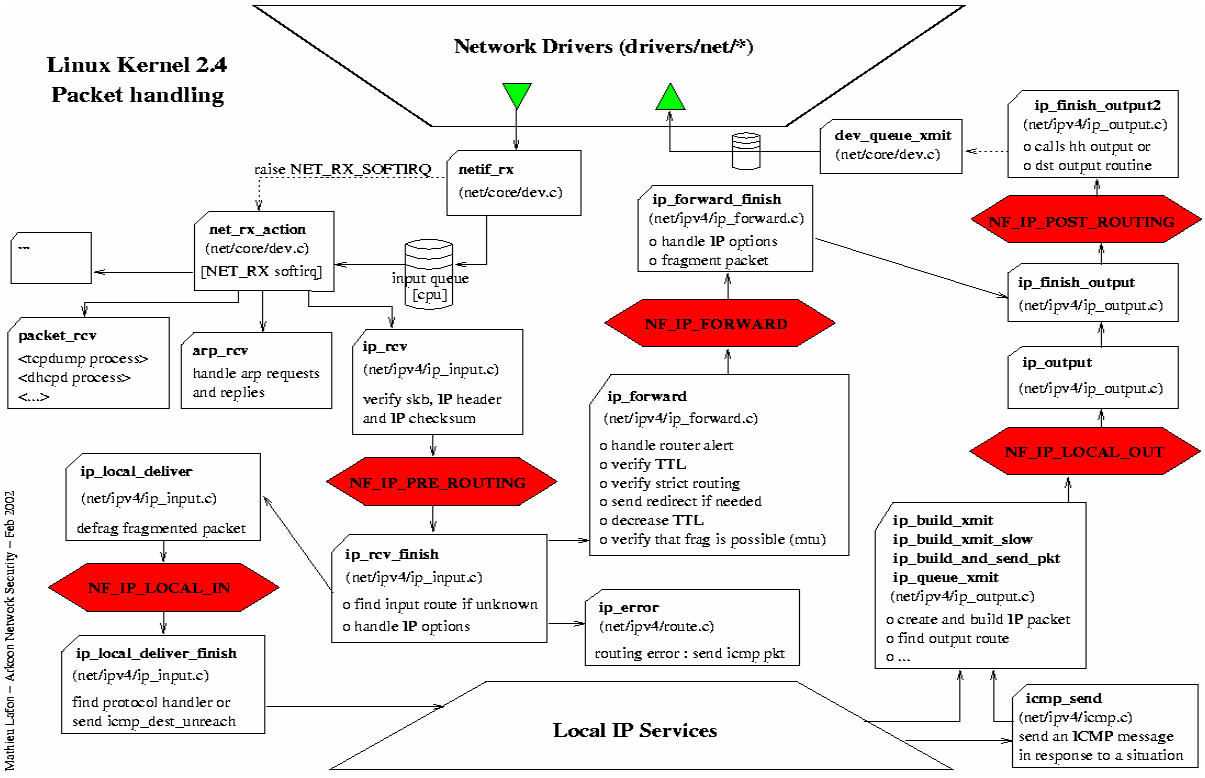

圖 1. 封包處理所呼叫的函式圖

圖 1[2]列出封包通過核心處理時所呼叫的函式圖，其中粗體黑線為函式名稱，圖中我們可以很清楚的看到封包在核心程式中被處理的流程順序，封包一共會經過 5 個 Netfilter framework 的 hook 點(圖中填滿的方塊)。作業系統核心的 netfilter function 會將流過 hook 點的網路封包傳給向其註冊的程式(user space program)，等待程式處理完封包後才繼續將封包往之後的函式做處理，而註冊的程式可以決定封包是否繼續往下傳、丟棄、做標記或是傳給其他程式。

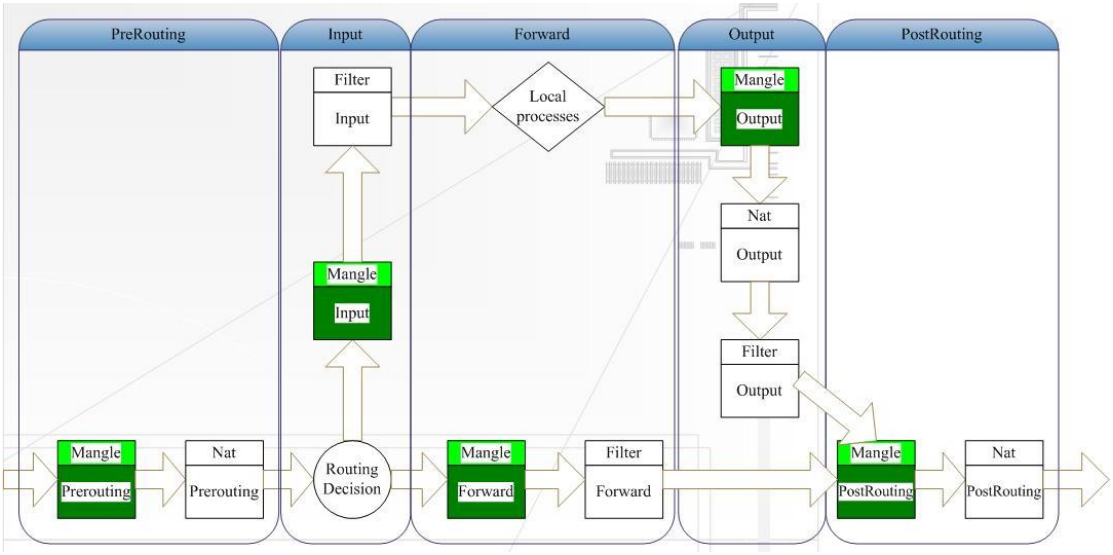

圖 2. IPTables 封包流程圖

圖 2 所示，這是封包流經 IPTables 各個表格(table)的流程圖，封包一開始會經過 Mangle 和 Nat 的 Prerouting，而中間的路徑(local process or forward)如何走則是由封包的目標位址來做決定，最後封包在離開 IPTables 前會經過 Mangle 和 Nat 的 Postrouting。

在實際的核心中，封包是通過 Netfilter 架構中的 hook，而因為 IPTables 有向 Netfilter 架構中的 hook 點作註冊動作，因此每個通過核心的網路封包便可以送到 IPTables，藉此來作使用者額外定義的封包過濾、修改動作。

IPTables 是一套能提供 NAT 與防火牆的軟體，目前內建於 Linux 2.4 核心中。其主要能夠根據來源位址、目的位址、網路介面、IP 通訊協定、是否為分段封包等多個項目來拒絕或允許封包的流入流出。除此之外 IPTables 還能：

- 為 TCP, UDP, ICMP 封包保留封包狀態資訊
- 提供重新導向 (redirection)
- 提供封包表頭給使用者指定的程式做認證

由於功能眾多，在本次介紹中，我們僅將重點放在和此次介紹套件 L7-filter 有關的 match modules 上。

2.1 IPtables 的規則與流程

IPTables 規則

```
//規則 1    利用 L7-filter 套件，將辯認到封包應用層 header 是 msn 的丟棄
iptables -t mangle -A INPUT -m layer7 --l7proto msnmessenger -j DROP
//規則 2    利用 L7-filter 套件，將辯認到封包應用層 header 是 http 所屬的做標記動作
iptables -t mangle -A POSTROUTING -m layer7 --l7proto http -j MARK --set-mark 1
```

表 1. IPTables 中使用 matching modules 的範例

基本上，IPTables 根據規則(rule)來描述允入或允出的封包條件。因此要使用 IPTables，便要懂得它規則的寫法，表 1 列出關於使用 matching modules 的規則及解釋，更近一步的介紹，請直接參考 IPTables HOWTO 網頁[3]。藉由加入”-m”選項，可以明確指定額外要載入的比對模組，藉此延伸擴充規則的比對功能。

IPTables 流程簡介

圖 3 是整個 IPTables 處理封包的流程圖，所有的步驟都是在 ipt_do_table() 這個函式中做的。

IPTables 主要步驟

1. 查看所在的 table 中是否還有規則未比較。如果已經沒有規則且封包未決定如何處理，則由所在 table 的政策(POLICY)作決定。
2. 檢查封包是否符合規則的 IP Header，呼叫 ip_packet_match()，並對照結構 ipt_entry 的內容做比較。
3. 檢查封包是否符合規則中的 extendsion match，呼叫 do_match()。
4. 增加通過此規則的封包計數器。
5. 檢查規則中是否有額外的 target function 來決定封包命運。

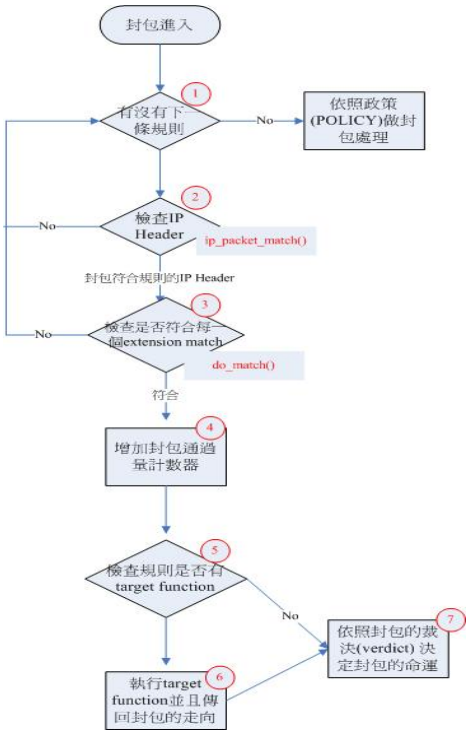

圖 3. IPTables 的流程圖

由圖 3 可以看出掛在 IPTables 上得 L7-filter 是在步驟 3 被執行的。

3. L7-Filter Classifier

在瞭解 IPTables 的規則與執行流程後，接下來介紹 L7-filter classifier，一個掛在 IPTables 中的 extendsion matching modules 的套件。關於此套件如何編譯安裝將不做敘述，有興趣可參考此套件的網站[1]。

本套件和其他 IPTables 的 extendsion matching modules 一樣，都是為了加強 IPTables 在分析封包的能力。如果一個 IPTables 規則上加上此 match modules(-m layer7 l7proto)，那麼就會多出一個結構為 ipt_entry_match 用來負責儲存 L7filter Classifier 套件的模組，因此 IPTables 在處理此封包時會看到封包的 OSI layer7 application layer 的資料，這表示我們將能更準確來分析封包。

3.1 L7-filter 運作方式

既然已經知道iptables會將封包交給L7-filter module來做處理。接下來將介紹此模組如何對封包進行辨別。L7-filter是使用V8 regular expressions [4] String-match的方式來辨識封包，其運作原理我們以ProFTP伺服器為例，當一個ftp客戶端要求連線後，ftp 伺服器會送一個含有*220 ProFTPD 1.2.9 Server*字串的封包表明自己，在客戶端輸入帳號後，伺服器會要求密碼而送一個含有*331 Password required for user*字串的封包。在歸納出眾多ftp伺服器每次送出的封包包含哪些字串後，我們將可以寫出一個ftp pattern file以供L7-filter module使用。那麼此模組將可藉由pattern file所提供的關鍵字來判斷網路封包是否為ftp協定的封包而做更強而有力的分析處理。

```
FTP
   ^220[\x09-\x0d -~]*ftp | 331[\x09-\x0d -~]*password
HTTP
   http/(0\.9|1\.0|1\.1) [1-5][0-9][0-9] [\x09-\x0d -~]*
(connection:|content-type:|content-length:)|post [\x09-\x0d -~]* http/[01]\.[019]
Msn-filetransfer
   ^ver [ -~]*msnftp\x0d\x0aver msnftp\x0d\x0ausr
```

表 2. pattern 的範例

表 2，我們舉出 ftp, http, man-filetransfer 的 pattern file。

FTP: 當封包的開頭是"220"之後有"ftp"字樣，並且之後的封包中包含"331"以及"password"字樣，則判斷此條連線是 ftp 應用。

HTTP: 當封包中包含"http", "connection:", "content-type:", "content-length:", "post"內容時，則判斷此條連線是 HTTP 應用。

MSN-filetransfer: 當封包開頭是"ver"，包含有"msnftp","aver","ausr"時，則判斷此條連線是 MSN-filetransfer 應用。

當然如果你發覺以上的寫法不妥或是想要對擁有某些特性的封包做處理，也是可以自己去撰寫 pattern file，預設是在/etc/l7-protocols/protocols 目錄

下會放應用層協定的 pattern file，目前支援的有 80 幾種應用協定[5]。

分類	應用程式名稱
Great	Bitorrent, FTP, hddtemp, HTTP
Good	AIM, Biff, CVS, DHCP, Direct Connect, DNS, Gkrellm, GnucleusLAN, Gnutella, Ident, IMAP, IP printing, IPC, MSN Filetransfers, MSN Messenger, Netbios name service, NCP, NNTP, POP3, Quake2, RTSP, Samba/SMB, SMTP, SNMP, SSH, Telnet, TSP, SSL, VNC, X Windows, Yahoo messenger
Ok	Apple, Audiogalaxy, BGP, Cisco VPN, Counter Strike, FastTrack, H.323, LPD, pressplay, RDP, rlogin, shoutcast, ZMAAP
Marginal	Goboogy, Jabber, live365, MUTE, OpenFT, Quake1, Skype, SOCKS, Subspace, Tesla, TFTP
Weak	eDonkey2000, Finger, Gopher, NetBIOS, WinMX

表 3. pattern 的分類

表 3，在目前支援的協定可以被區分為：Great, Good, Ok, Marginal, Weak 等五種 pattern file，會將 pattern file 區分成五類主要是因為有些協定(如 HTTP,FTP)是公開的，此類的連線封包內容的關鍵字是完全可以掌握的，因此寫出來的 pattern 當然是大家皆可使用；但是仍然有些協定是不公開的，對於此類的協定只能藉由抓取其封包內容，並且觀察封包會出現的關鍵字來做 pattern，所以此類的 pattern 可能對於別的使用者來說是會發生錯誤的判斷。對於運作效果不是很好的便會歸於 marginal 或 weak 這兩類。

3.2 L7-filter pseudo code

```
int match(packet, protocol)
{
        if(regular expression for the protocol is not compiled yet)
                compile it and put it in a list of compiled regexps;
        else
                fetch the compiled pattern from the list;

        if(already classified this connection)
                if(classification matches one we're looking for)
                        return true;
                else
                        return false;

        if(seen too many packets with no match)
                return false;

        Append application layer data to data buffer;

        if(data buffer matches regexp for the protocol we're looking for)
                Mark the connection as identified;
                return true;
        else
                return false;

}
```

圖 4. L7-filter pseudo code

圖 4[6]，是L7-filter套件的pseudo code，在match函式中傳入封包*packet*以及要比較的*protocol*，此套件程式碼共分四個區塊，

1. 檢查要比較的*protocol*是否已經放到compiled regexps的list中。

   因為 L7-filter 套件預設寫好的 pattern file 有 80 多個， 而使用者在其 IPTables 中會使用到的 pattern 可能只是其中幾個； 並且當使用者定義好 IPTables 後，通常不再會有變更，因此套件只會將需要用到的 pattern 做編譯並且放入 list 中，而不是將所有的 patten 都放入比較的 list 內。

2. 檢查此連線是否已被識別過，不是的話則繼續往下做。

   如果是的話則會看現在要比較的*protocol*和之前連線判斷出來的protocol是否一樣，再依照結果傳回*packet*和*protocol*是否符合(match)。

3. 此連線已檢查的封包數是否超過界定範圍(預定範圍是 8 個封包)。

   若檢查封包數超過 8 個則直接回傳*packet*和*protocol*是不符合的。如此做是因為通常看連線前面幾個封包便能判斷出連線是屬於哪個協定；另外如此做的話只需要看前面少數封包，而不用看連線上的所有封包，這對於判斷的效率上來說有莫大的幫助。

4. 用一個 data buffer(大小 4kb)來儲存此連線的封包(前 8 個)，並且看 data buffer中是否有出現關鍵字(在pattern中定義的)，如果有的話則將此連線判定為符合*protocol*。

4. 相關套件介紹（IPP2P）

除了本文主要介紹的 L7-filter Classifier 套件外，另外尚有一個套件 IPP2P[7]，也能分析封包到 OSI layer 7。由名稱大概可以了解此套件的功用在於分析封包是否為 p2p 軟體所發送，此套件的運作方式和 L7-filter Classifier 方式相似，都是掛在 IPTables 的 extendsion matching modules 上(使用選項

-m)、而且同樣是使用 String-match 的方式，不同的是他們在 String-match 上的比較方法不同，L7-filter 是有一個額外的 pattern file 檔案，每次要比對時便去和此 pattern file 做比對，然而 ipp2p 則是將 String-match 的內容直接寫入程式碼中之後再被編到模組內，因此 ipp2p 所支援應用軟體辨別的數目上自然比較少。

以下讓我們做個小測試實驗，針對這二套軟體（ipp2p 與 L7-filter）稍作比較。

圖5. Max. throughput和packet size關係

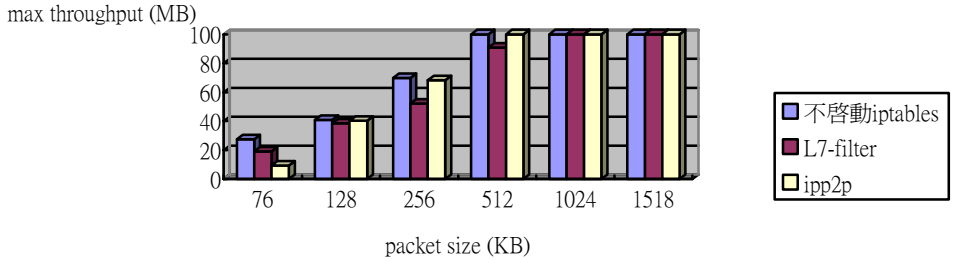

圖6. Average latency和packet size的關係

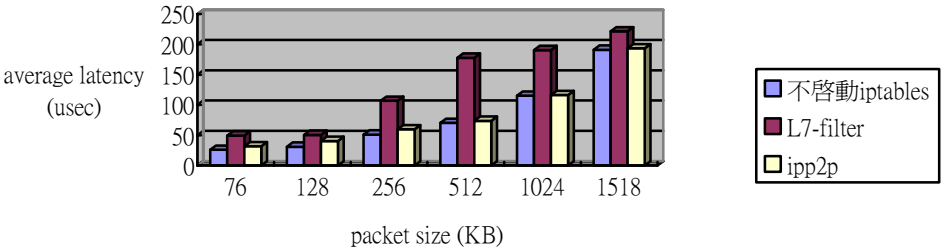

圖 5,6 是將兩套套件裝在同一部主機做的測試，測試項目分別有

(1) 不啟動 iptables

(2) 使用 l7-filter，並且加入一行規則

iptables -t mangle -A POSTROUTING -m layer7 --l7proto edonkey -j DROP

(3) 使用 ipp2p，並加入一行規則

Iptables –t mangle –A POSTROUTING –m ipp2p –edk –j DROP

測試的主要目的是看兩套套件在單獨加入一行指令只擋住 edonkey 封包時，三種情況的效能差異，圖 5 觀察在不同 packet size，各自最大處理力(throughput)。圖 6 則是在觀察不同 packet size 下，各自延遲時間是多少。

由上面實驗結果可以清楚的看出兩個套件總體來說，ipp2p 效能比較好，但

是因為 ipp2p 的比較 pattern 是寫入程式碼中，因此在使用上不如 L7-filter 方便。表 4 做兩者間的比較總結。

	L7-filter	IPP2P
功能	Filter, Qos	Filter
運作方式	iptables/netfilter, Tc	iptables/netfilter
封包辨別方式	String match	String match
辨別的方法	Using pattern file	Write in code
是否允許使用者定義辨別條件	yes	no
最大生產力(throughput)	較差	較好
延遲時間 (Latency)	較長	較短
目前可辨別的軟體協定封包數目	82	6

表 4. 兩套件比較

5. 結論

為了增加資訊安全或是服務品質，有時我們必須在路由器上做封包的辨別與處理，傳統上針對傳輸層(OSI layer4)做查看得知封包屬於何種軟體的判別方式因為近年來 dynamic-port application(如 p2p 軟體)的日益增多，一個軟體可開啟未知的 port 來做傳輸，這對於封包的辨識上莫過於是一大挑戰，在此次報告我們針對此等問題利用 L7-filter classifier 套件來解決。另外，此套件除了可做 filter 功能外，也可以有 QoS 的功能，其運作方式則是掛於 tc 下，其實際細節因為超出本篇報告範圍因此不做說明，若想深入了解，可到網站上去查詢。

6. 參考資料

[1] L7-filter classifier: http://l7-filter.sourceforge.net/

[2] 圖 2. 擷取自 http://open-source.arkoon.net/kernel/kernel_net.png

[3]IPTables HOWTO:

http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html#toc6

[4] V8 regular expressions:

http://www.hmug.org/man/3/regsub.html

[5] L7-filter supported protocols

http://l7-filter.sourceforge.net/protocols

[6] 圖 4 擷取自http://l7-filter.sourceforge.net/technicaldetails

[7]IPP2P： http://rnvs.informatik.uni-leipzig.de/ipp2p/index_en.html