Linux 2.4 NAT HOWTO

作者：Rusty Russell, mailing list netfilter@lists.samba.org
譯者：網中人 netmanforever@yahoo.com v1.0.1 Mon May 1 18:38:22 CST 2000

本文件描述如何用 2.4 Linux 核心去做 masquerading transparent proxying port forwarding 和其它類型的 Network Address Translations。

目錄

1 Introduction

親愛的讀者 ，歡迎您！

您將要探索的是引人入勝(有時蠻恐怖)的 NAT(Network Address Translation) 世界 ，同時 ，您甚至可以把這份 HOWTO 當成 Linux 2.4 核心及以後版本的精確指南呢。

在 Linux 2.4 裡面 ，有一個叫 'netfilter' 的東東 ，它是專門撕裂 (mangling*)封包的。在它再上一個層級 ，就是提供 NAT 功能的了 ，則是完全由以往的核心實作而成的。

(譯者註：很奇怪 ，原作者用 mangle 這一詞 ，似乎在過往的中文文件中都沒碰到過 ，查過好多字典都不知道怎麼翻譯好。這裡暫時勉強用‘撕裂’這個詞代替 ，不過後面我就不嘗試翻譯這詞了 ，讓讀者自己去理解吧。)

2 官方的網站和通信論壇何處覓？

目前有三個官方網站可供瀏覽：

- 感謝 *Filewatcher (http://netfilter.filewatcher.org)* `<http://netfilter.filewatcher.org>`.
- 感謝 *The Samba Team and SGI (http://www.samba.org/netfilter)* `<http://www.samba.org/netfilter>`.
- 感謝 *Jim Pick (http://netfilter.kernelnotes.org)* `<http://netfilter.kernelnotes.org>`.

而官方的 netfilter 郵件論壇 ，則可以到這裡看： *Samba's Listserver (http://lists.samba.org)* `<http://lists.samba.org>`.

2.1 何為 Network Address Translation?

一般來說 ，在網路上封包從其來源(比方您家中的電腦)出去 ，然後到達目的地(比方www.kernelnotes.org) ，會經過許許多多個不同的連接(links)：就我所在的澳洲來說就大約有 19 個之多。沒有任何一個連接會真的去更改您的封包：他們僅僅是將之傳送出去而已。

假如其中一個連接會做 NAT 的話 ，然後它們就會更改那些經它而過的封包之來源或目的地地址。誠如您能想像得到的 ，這並非系統被設計成這樣的 ，而是 NAT 所做的手腳而已。通常要做 NAT 的連線

會記住它如何 mangled 封包的，然後當回應封包從另一方向過來的時候，然後就反過來 mangling 那個回應封包，所以所有東西都工作起來了。

2.2 為什麼我要做 NAT 呢？

在完美的世界裡，您無需這樣做啦。在目前來說，還是有其理由的：

用 **modem** 撥接上網

大多數的 ISP 在您連上去的時候只會給您一個單一 IP 地址。您喜歡的話，以任何來源地址把封包送出去都行，但只有回應到這個來源地址的封包才可以回到您那裡。如果您想用多台不同主機(例如家中網路)透過該連接上 internet 的話，那您就要 NAT 了。

這也就是今天 NAT 最常用之處，而在 Linux 世界最為人知的就是所謂的 'masquerading(封包偽裝術)' 了。我稱之為 SNAT，因為您改變了第一個封包的 **source(來源)** 地址的緣故。

多重伺服器

有時候，您會想去改變那些進入您網路的封包之路向。這最常是因為(如上述)您只有一個 IP 地址，但您卻想讓別人能夠連接到 '真實' IP 地址後面的主機去。如果您重寫這些內送封包的目的地址，這樣您就可以管理它們了。

一個常見的變動是負載分擔(load-sharing)，也就是在一組機器上面為封包做映對(mapping)的動作。這類型的 NAT，在以前的的 Linux 版本中也就被稱為 port-forwarding。

透明代理**(Transparent Proxying)**

有時候，您或許想要每一個經過您的 Linux 主機的封包送至主機本身的一個程式去。這就需要進行透明代理的動作了：一個代理就是一個位於您的網路和外部網路的程式，為彼此雙方負起溝通的任務。而所謂的透明，則是因為您的網路甚至無須知道在和一個代理對講，當然了，除非代理不再工作了吧。

Squid 可以配置成這樣的工作方式，這就是在過往的 Linux 版本中所謂的重導向(redirection)或透明代理了。

3 兩種類型的 NAT

我將 NAT 分為兩種不同的類型： **Source NAT** (SNAT) 與 **Destination NAT** (DNAT)

Source NAT 就是您將改變第一個封包的來源地址：例如，您為傳入的連線做 caching 的動作。Source NAT 永遠會在封包傳出網線之前就做好 post-routing 的動作。封包偽裝(Masquerading)就是一個 SNAT 特例。

Destination NAT 就是您將改變第一個封包的目的地地址：例如您要為傳出的連線做 caching 的動作。Destination NAT 永遠會在封包從網線進入之後就馬上做好 pre-routing 的動作。Port forwarding負載分擔以及透明代理，都屬於 DNAT。

4 從 2.0 到 2.2 核心的快速轉變

非常抱歉，假如您仍然忙於從 2.0(ipfwadm) 到 2.2(ipchains) 的轉型的話。不過，這也是個喜憂參半的消息啦。

首先，您可以輕易的一如往昔地使用 ipchains 和 ipfwadm。要這樣做的話，您需要將最新的 netfilter 套件中的 'ipchains.o' 或 'ipfwadm.o' 核心模組載入。它們是相互排斥的(您應已獲警告了)，同時也不能和其它 netfilter 模組同時整合在一起。

一旦其中一個模組被載入，您就可以如常使用 ipchains 和 ipfwadm 了，但也有如下一些變化啦：

- 用 ipchains -M -S，或是 ipfwadm -M -s 作偽裝逾時將不再有效。因為逾時設定已經移至新的 NAT 架構中，所以這裡也就沒什麼所謂了。
- 在偽裝列表中顯示的 ini_seqdelte和 previous_delta 欄位，將永遠為零。
- 同時歸零(zeroing)和列示記數器(counter)的 '-Z -L' 已無作用：記數器將不能再歸零了。

Hacker 們仍要留意之處：

- 您現在可以捆綁 61000-65095 之間的埠口，而無需理會您是否使用封包偽裝技術。在過去，封包偽裝程式會把此值域內的所有東西捕獲進來，所以其它程式就不可用之了。
- 至於(尚未成文之) getsockname 破解，在過去，透明代理程式可以找出那些不再有效連線之真正目的地。
- 至於(尚未成文之) bind-to-foreign-address 破解，同樣尚未實作；這在過去用以完善透明代理的構想。

4.1 救命啊！我只想要封包偽裝而已！

沒錯，這也是大多數朋友之需。如果您用 PPP 撥接獲得的動態 IP (如果您不了解的話，那您應該是了)，您或許只想單純告訴您的主機讓所有來自您內部網路的封包，看起來如來自該 PPP 撥接主機一樣。

```
# Load the NAT module (this pulls in all the others).
modprobe iptable_nat

# In the NAT table (-t nat), Append a rule (-A) after routing
# (POSTROUTING) for all packets going out ppp0 (-o ppp0) which says to
# MASQUERADE the connection (-j MASQUERADE).
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

註：您這裡並沒做任何封包過濾：如要的話，請參考 Packet Filtering HOWTO：將 NAT 和封包過濾合併起來就是了。

4.2 那 **ipmasqadm** 怎麼了？

這個其實取決於使用者而已，所以我並不是很為向後兼容問題而擔心。您可以單純使用 iptables -t nat 做 port forwarding 的動作。例如，在 Linux 2.2 您或許已經這樣做了：

```
# Linux 2.2
# Forward TCP packets going to port 8080 on 1.2.3.4 to 192.168.1.1's port 80
ipmasqadm portfw -a -P tcp -L 1.2.3.4 8080 -R 192.168.1.1 80
```

而現在 ，如此則可：

```
# Linux 2.4
# Append a rule pre-routing (-A PREROUTING) to the NAT table (-t nat) that
# TCP packets (-p tcp) going to 1.2.3.4 (-d 1.2.3.4) port 8080 (--dport 8080)
# have their destination mapped (-j DNAT) to 192.168.1.1, port 80
# (--to 192.168.1.1:80).
iptables -A PREROUTING -t nat -p tcp -d 1.2.3.4 --dport 8080 \
        -j DNAT --to 192.168.1.1:80
```

假如您想讓這條規則同時修改本機連線的話(如 ，即使在 NAT 主機本身 ，要連接 1.2.3.4 的 8080 埠口之 telnet 連線 ，會幫您連接至 192.168.1.1 的 80 埠口) ，您就可以插入相同的規則至 OUTPUT 鏈中(它只適用於本機傳出的封包)：

```
# Linux 2.4
iptables -A OUTPUT -t nat -p tcp -d 1.2.3.4 --dport 8080 \
        -j DNAT --to 192.168.1.1:80
```

5 控制哪些要 NAT

您需要建立一些 NAT 規則 ，來告訴核心哪些連線要改變 ，同時如何去改變它們。要做到這點 ，我們需要一個非常多用途的 `iptables` 工具 ，同時指定 '-t nat' 選項告訴它去修改 NAT 表格。

NAT 規則的表格含有三個列表叫做'chains'：每一條規則都按順序檢查 ，直到找到一個相符的比對。該三個鏈就叫做 PREROUTING (對 Destination NAT 來說 ，因為封包首先是傳入的)POSTROUTING (對 Source NAT 來說 ，因為封包是離開的)以及 OUTPUT (對 Destination NAT 來說 ，是指那些由本機產生的封包)。

假如我夠藝術天份的話 ，下面的圖示將準確模擬出上面所說的概念。

```
      -----                                       -----
    /       \                                   /       \
  PREROUTING -->[Routing ]----------------->POSTROUTING----->
    \D-NAT/     [Decision]                     \S-NAT/
                    |                             ^
                    |                           __|__
                    |                         /       \
                    |                        | OUTPUT|
                    |                         \D-NAT/
                    |                             ^
                    |                             |
                    --------> Local Process ------
```

於前述的每一點 ，當一個封包通過我們要查看的相關連線之時 ，如果它是一個新建連線 ，我們查看它在 NAT 表格裡對應的鏈 ，看看能對之做些什麼動作。而由此獲得的答案就應用於該連線將來的所有封包。

5.1 用 iptables 做簡單的選擇

`iptables` 具有如後所列的許多標準選項。所有那些帶雙減號的選項都是可以縮寫的，只要 `iptables` 仍可將之與其它可能的選項區分開來就行。如果您的核心以模組形式來支援 iptables，您就需要首先載入 ip_tables.o：'insmod ip_tables'。

這裡，最重要的一個選項是表格選擇選項：'-t'。對於所有的 NAT 操作，您會想用 '-t nat' 來表示 NAT 表格。第二個重要的選項是以 '-A' 增加一條新規則至鏈的末端 (如：'-A POSTROUTING')，或以 '-I' 插入至前端(如：'-I PREROUTING')。

您可以指定您要做 NAT 的封包來源地址 ('-s' 或 '--source') 與目的地 ('-d' or '--destination')。這兩個選項後面可以後接一個單一的 IP 地址 (如：192.168.1.1)，或一個名稱 (如：www.kernelnotes.org)，或一個網路地址 (如：192.168.1.0/24 或 192.168.1.0/255.255.255.0)。

您也可以指定要比對的傳入 ('-i' 或 '--in-interface') 和傳出 ('-o' or '--out-interface') 界面，但哪一個界面可以指定則取決於您要將規則寫入哪一個鏈去：對於 PREROUTING，您可以選擇傳入界面，但對於 POSTROUTING (以及 OUTPUT)，您可以選擇傳出界面。如果您不小心用錯了，`iptables` 就會給您一個錯誤。

5.2 關於挑選哪些封包來 mangle 的細節

我前面已經說過，您可以指定來源和目的地地址。如果您省略來源地址的選項，那麼就泛指任何來源。如果您省略目的地地址，則泛指所有目的地地址。

您還可以指定一個特定協定 ('-p' or '--protocol')呢，例如 TCP 或 UDP：只有這些協定的封包才符合該規則。其主要原因是，指定 tcp 或 udp 協定可以允許更多選項：尤其是 '--source-port' 與 '--destination-port' 選項 (縮寫為 '--sport' 與 '--dport')。

這些選項可以讓您指定只有哪些特定來源和目的地埠口的封包才符合該規則。這在您要重導 web 請求 (TCP port 80 或 8080) 但又怕影響其它封包的時候，就很好用了。

這些選項必須接在 '-p' 選項的後面(這會在為該協定載入共享函式庫時有副作用)。您可以使用埠口號碼，或者是在 /etc/services 檔中的名稱。

所有這些您能選擇的封包之不同品質，都詳細列在那個詳細得有點恐怖的 manual page 中了(`man iptables`)。

6 談談要怎樣 Mangle 封包

現在，我們知道如何去挑選那些我們要 mangle 的封包。為了要完善我們的規則，我們需要準確無誤的告訴核心，什麼才是我們要對封包做的。

6.1 Source NAT

您想要做 Source NAT，是要去將連線的來源地址換成別的什麼的。這就要在它最後要送出去之前，於 POSTROUTING 鏈中完成了；這是一個非常重要的細節，因為它意味著所有在 Linux 主機本身上的其它東西 (routing, packet filtering) 都只看見那個還沒改變的封包。同時，這也就是說，'-o' (傳出界面) 選項可以派上用場了。

Source NAT 是用 '-j SNAT' 來指定的 ，同時 ，'–to source' 則指定一個 IP 地址或一段 IP 地址以及一個可配選的埠口或一段值域的埠口(僅適用於 UDP 和 TCP 協定)。

```
## Change source addresses to 1.2.3.4.
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4

## Change source addresses to 1.2.3.4, 1.2.3.5 or 1.2.3.6
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.6

## Change source addresses to 1.2.3.4, ports 1-1023
# iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to 1.2.3.4:1-1023
```

6.1.1 封包偽裝 (Masquerading)

有一個 Source NAT 之特例 ，叫做封包偽裝：它只用於動態分配的 IP 地址 ，例如標準的撥接(如果用靜態 IP 地址 ，則使用前述之 SNAT)。

您無需明確地將 masquerading 放進來源地址那裡去：它將會使用封包傳出界面作為來源地址。但更重要的是 ，如果該連接(link)斷掉的話 ，那麼連線 (connections ，無可避免的將失掉) 也會被忘掉 ，當連線用新的 IP 地址回來的時候就會有問題了。

```
## Masquerade everything out ppp0.
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

6.2 Destination NAT

一旦封包進入 ，會由 PREROUTING 鏈完成處理；也就是說 ，除了該主機自己的其它東西(諸如：路由封包過濾) 都將封包看成要送到 '真正' 目的地。另外 ，那個 '-i' (傳入界面) 選項也可以在這裡使用。

需要修改本機產生的封包之目的地的話 ，那麼 OUTPUT 鏈就可以用上了 ，不過這並不常碰到。

Destination NAT 必須以 '-j DNAT' 來指定使用 ，同時用 '–to destination' 選項指定一個 IP 地址或一段 IP 地址 ，以及可以配選一個埠口或一段埠口值域(只能用於 UDP 和 TCP 協定上面)。

```
## Change destination addresses to 5.6.7.8
# iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 5.6.7.8

## Change destination addresses to 5.6.7.8, 5.6.7.9 or 5.6.7.10.
# iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 5.6.7.8-5.6.7.10

## Change destination addresses of web traffic to 5.6.7.8, port 8080.
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 \
        -j DNAT --to 5.6.7.8:8080

## Redirect local packets to 1.2.3.4 to loopback.
# iptables -t nat -A OUTPUT -d 1.2.3.4 -j DNAT --to 127.0.0.1
```

6.2.1 重導向 (Redirection)

在 Destination NAT 有一個特別的情形：它是一個簡單的便利，完全等同於給傳入界面地址做 DNAT 一樣。

```
## Send incoming port-80 web traffic to our squid (transparent) proxy
# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
        -j REDIRECT --to-port 3128
```

6.3 進一步的映對(Mappings)

還有許多 NAT 上面的解決方案是大多數人無需用到的。這裡不妨和那些有興趣的朋友探討一下：

6.3.1 同一範圍內的複合地址(Multiple Addresses)之選擇。

如果您已經指定了一段 IP 地址，而 IP 地址的使用選擇是基於機器所知連線目前最少使用之 IP。它可以提供最原始的平衡負載(load-balancing)。

6.3.2 建立空 NAT 映對

您可以使用 '-j ACCEPT' 目標來讓一個連線通過，而繞過 NAT 的處理。

6.3.3 標準的 NAT 行為(Behaviour)

預設的行為是在使用者制定的規則限制內，盡可能少的改變連線。換而言之，非不得已不要重映對(remap)埠口。

6.3.4 絶對來源埠口映對

如果其它連線已經被映對到新的連線，就算對於一個無需 NAT 的連線來說，來源埠口的轉換有時或是必須絶對存在的。讓我們假設一個封包偽裝的情形，這已經非常普遍了：

1. 一個網頁連線由一台 192.1.1.1 的機器從 port 1024 建立，要連接到www.netscape.com port 80。
2. 它被封包偽裝主機以其自己的 IP 地址(1.2.3.4)進行偽裝。
3. 該封包偽裝主機嘗試由 1.2.3.4 (它的外部界面地址) port 1024 來做一個網頁連線至www.netscape.com port 80。
4. 然後 NAT 程式改變第二個連線的來源埠口為 1025，所以這兩個連線不至於相衝(clash)。

當這個絶對來源映對存在之時，埠口被拆分為三個等級：

- 512 以下的埠口
- 512 到 1023 之間的埠口
- 1024 以上的埠口

任何一個埠口都不會被絶對映對到不同的等級去。

6.3.5 當 NAT 失效時會怎樣？

如果沒有辦法如用戶要求那樣獨一無二地映對連線 ，那麼連線就會被擋掉。當一個封包不能夠界定為任何連線的時候 ，結果也一樣 ，因為它們可算是畸形的 ，或者是該機器記憶體耗光了 ，諸如此類。

6.3.6 複合映對重疊和相衝(clash)

您可以設定 NAT 規則在同一個範圍之上映對封包；NAT 程式足以聰明的去避免相衝。比方說 ，用兩條規則將 192.168.1.1 和 192.168.1.2 這兩個來源地址分別映對到 1.2.3.4 ，是完全可行的。

再來 ，您可以映對到真實的已用的 IP 地址 ，只要這些地址通過這個映對主機就行。所以 ，如果您獲得一個網路(1.2.3.0/24) ，但有一個內部網路使用這些地址 ，而另一個使用私有地址 192.168.1.0/24 ，您就可以 NAT 那些 192.168.1.0/24 的來源地址到 1.2.3.0 網路之上 ，而無需擔心相衝：

```
# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 \
        -j SNAT --to 1.2.3.0/24
```

這同樣適用於那些 NAT 主機自己使用的地址：這其實就是封包偽裝如何工作的了(分享偽裝封包地址和來自主機本身封包之 '真實' 地址。)

更甚者 ，您還可以映對相同的封包到許多不同的目標(targets)上去 ，而且它們都是共享的。例如 ，如果您不想映對任何東西到 1.2.3.5 上去 ，您可以這樣做：

```
# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 \
        -j SNAT --to 1.2.3.0-1.2.3.4 --to 1.2.3.6-1.2.3.254
```

6.3.7 改變本機產生的連線之目的地

如果本機產生的封包之目的地改變了(例如 ，用 OUTPUT 鏈) ，而這樣會導致封包由不同的界面送出去 ，這樣來源地址也跟著變為那個界面。舉例子說 ，改變一個環迴(loopback)封包之目的地由 eth0 送出 ，會讓來源地址也由 127.0.0.1 變成 eth0 的地址；而不像其它來源地址映對那樣 ，這是立即完成的。當然 ，所有這些映對在回應封包進入時是顛倒過來的。

7 特殊協定

有些協定是並不想要做 NAT 的。對於每一個這樣的協定而言 ，有兩個延伸設定(extension)是必須要寫清楚的：一個是關於協定之連線追蹤 ，另一個關於實際的 NAT。

在 netfilter 發行套件裡面 ，有一些關於 ftp 的現行模組：ip_conntrack_ftp.o 與 ip_nat_ftp.o。如果您把這些插入到您的核心裡面(或您永久性的編譯它們) ，那麼要在 ftp 連線上做任何種類的 NAT 都是可行的。如果您不這樣的話 ，那您可以使用被動模式 ftp ，不過如果您要做一些動作甚於簡單 Source NAT 的話 ，這就可能不那麼可靠了。

8 NAT 的一些限制 (caveats)

如果在一個連線上做 NAT ，所有 雙向 (傳出和傳入) 的封包 ，都必須要通過 NAT 主機才行 ，否則並不可靠。尤其在連線追蹤程式重組碎片 (fragments)的時候 ，也就是說 ，不但連線追蹤會不可能 ，而且您的封包根本就不能通過 ，因為碎片會被擋下。

9 Source NAT 與路由

如果您要做 SNAT ，您會想要確定經過 SNAT 封包所傳給的主機會將回應送回給 NAT 主機。例如 ，如果您映對某些傳出封包到來源地址 1.2.3.4 之上 ，那麼外部的路由器就必須知道要將回應封包(目的地為 1.2.3.4)送回給該主機。這可以用如下方法做到：

1. 如果您要在主機自己的地址(路由和其它所有運作皆正常)上面做 SNAT ，您無需做任何動作。
2. 如果您要在一個在本機網路上尚未使用的地址做 SNAT(例如 ，映對到在 1.2.3.0/24 網路上的一個可用 IP 1.2.3.99) ，您的 NAT 主機就需要回應關於該地址的 ARP 請求 ，一如它自己本身的一樣：最簡單的方法就是建立 IP alias ，例如：

   ```
   # ip address add 1.2.3.99 dev eth0
   ```

3. 如果您要在一個完全不同的地址上做 SNAT ，您就要確定 SNAT 封包抵達的機器能夠路由回該 NAT 主機。如果 NAT 主機是它們的預設閘道器的話 ，是可以做到的 ，否則 ，您就要廣告(advertize)一個路由(如果跑路由協定的話) ，或是手工的在每一台參與機器上增加路由。

10 在同一網路上的 Destination NAT

如果您要做 portforwarding 回到同一個網路 ，您要確定前向和回應封包雙方都經過該 NAT 主機(這樣它們才能被修改)。NAT 程式從現在開始(2.4.0-test6以後) ，會擋掉後面情形所產生的傳出 ICMP 重導向：那些已經 NAT 的封包以它所進入的相同界面傳出 ，而接收端伺服器仍嘗試直接回應到客戶端(不認可該回應)。

經典的情形是內部人員嘗試連接到您的 ‘公有(public)’ 網站伺服器 ，實際上是從公有地址(1.2.3.4) DNAT 到一個內部的機器(192.168.1.1)去 ，就像這樣：

```
# iptables -t nat -A PREROUTING -d 1.2.3.4 \
        -p tcp --dport 80 -j DNAT --to 192.168.1.1
```

一個方法是跑一台內部 DNS 伺服器 ，它知道您的公有網站的真正(內部) IP 地址 ，而將其它請求轉傳給外部的 DNS 伺服器。換而言之 ，關於您網站伺服器的記錄會正確地顯示為內部 IP 地址。

而另一個方法是同時讓這台 NAT 主機將該等連線之來源 IP 地址映對為它自己的地址 ，我們可以像如下那樣做(假設 NAT 主機之內部 IP 地址為 192.168.1.250)：

```
# iptables -t nat -A POSTROUTING -d 192.168.1.1 -s 192.168.1.0/24 \
        -p tcp --dport 80 -j SNAT --to 192.168.1.250
```

因為 **PREROUTING** 規則是最先執行的 ，對內部網站伺服器而言 ，封包就已經被定向好了：我們可以內定好哪個為來源 IP 地址。

11 感謝

首先感謝在我工作期間相信 netfilter 的構想並支持我的 WatchGuard 和 David Bonn。

以及所有其他幫我指正 NAT 之不足的朋友 ，尤其是那些讀過我的日記的。

Rusty.