Asas pendinding api : *netfilter/iptables* versi 1.0.3

Mohammad Bahathir Hashim
bahathir@gmail.com

January 22, 2008

Abstract

iptables adalah perisian untuk mengawal modul pendinding api binadalam kernel Linux versi 2.4 dan keatas yang dipanggil *netfilter*. Untuk memahami penggunaan *iptables*, pemahaman mengenai pendinding api (*firewall*) dan asas *TCP/IP* amat digalakkan. Walaubagaimana pun artikel ini akan menghuraikan konsep asas pendiding api yang juga boleh diguna pakai pada mana-mana aplikasi pendinding api lain.

Diharapkan artikel ini dapat memberi sedikit sebanyak ilmu asas bagi memulakan pembianaa pendinding api. Sila kongsikan artikel ini kepada rakan-rakan lain, agar ilmu ini dapat disebarkan. Seperti biasa segala kandungan artikel ini adalah sumber terbuka dibawah lesen GPL versi 2.

1 Pengenalan

Pendinding api adalah salah satu cara bagi meningkatkan lagi tahap keselamatan, dan secara tidak langsung akan mengurangkan risiko sistem jaringan dari dicerobohi. Kefahaman pendinding api sangat penting bagi memastikan ianya dapat berfungsi menurut kehendak kita dan mengurangkan perkara yang tidak dijangka akan terjadi. Walaupun terdapat pelbagai jenis perisian dan perkakasan pendinding api dalam dunia ini, tetapi sekiranya ianya salah dipasang (*configure*), maka fungsinya bagi melindungi jaringan sudah tiada ertinya lagi. Harus diingat, bahawa pendinding api hanyalah salah satu cara bagi mempertingkat ciri-ciri keselamatan, dan ianya perlu ditambah dengan teknik lain lagi bagi memastikan sistem dan jaringan kita lebih selamat.

Walaupun terdapat pelbagai jenis perisian dan skrip yang dapat membantu memudahkan kita membina polisi ini, tetapi, pengetahuan asas yang mendalam adalah lebih diutamakan, kerana dengan pengetahuan ini, anda boleh membina polisi walaupun pada konsol teks (CLI) dengan resources yang minima.

2 Asas TCP/IP

Oleh kerana pendinding api adalah cara untuk menapis paket dalam jaringan **TCP/IP**, maka adalah lebih baik dimulakan dengan asas jaringan **TCP/IP**. Secara ringkas, jaringan **TCP/IP** memerlukan informasi mengenai alamat punca (*source address*) bersama *port*nya (*source port*) dan alamat penerima (*destination address*) *port*nya (*destination port*). Buat masa ini, mari kita tumpukan kepada 4 parameter ini terlebih dahulu. Didalam artikel ini, saya akan ringkaskannya dalam bentuk dibawah.

$$SRC : SPT \longrightarrow DST : DPT$$

- **SRC** : Alamat punca (*source address*)
- **SPT** : Port punca (*source port*)
- **DST** : Alamat penerima (*destination address*)
- **DPT** : Alamat punca (*destination port*)

Contoh:
Saya ingin membuat sambungan *http* ke B *google.com* dari komputer saya A yang mempunyai alamat IP *162.142.2.17*.

Dari ayat diatas, mari kita asingkan dan ringkaskan kepada bentuk atas. SRC adalah *162.142.2.17* yakni alamat IP yang mengirim atau memulakan sambungan kepada DST (*google.com*) pada port DPT *http* (80). Disini kemana perginya port punca SPT? Biasanya, sistem akan menggunakan port yang dipilih secara rawak. Port sebegini biasanya lebih tinggi dari nilai 1024. Bagi memudahkan contoh, saya tetapkan SPT sebagai 1721.

$SRC = 162.142.2.17$
$SPT = 1721$

$DST = google.com$
$DPT = 80$

$162.142.2.17 : 1721 \longrightarrow google.com : 80$

Ini baru, sambungan bagi menghantar permintaan kepada *google.com*. Biasanya, *google.com* akan membalas balik dengan hasil keputusan kepada kita. Di takat ini, *google.com* akan menjadi pengirim atau punca sambungan ke komputer saya. Secara gambaran ianya akan jadi sebegini.

$SRC = google.com$
$SPT = 80$

$DST = 162.142.2.17$
$DPT = 1721$

$google.com : 80 \longrightarrow 162.142.2.17 : 1721$

Dari contoh ini, jelas bahawa kebanyakkan sambungan jaringan *TCP/IP* adalah hubungan 2 hala, yakni sambungan penghantaran permintaan (*request*) dan hasil (*reply*) dari permintaan tadi akan melalui sambungan yang berbeza. Kalau saya gabungkan kedua-dua gambaran diatas ianya akan jadi macam ini. Arah pergerakan merupakan kunci utama bagi menentukan yang mana **SRC:SPT** dan **DST:DPT**. Ini dapat diringkaskan sebegini.

$$A \underset{jawab}{\overset{hantar}{\rightleftarrows}} B$$

Kefahaman yang tinggi mengenai perkara ini akan membantu pemahaman kita mengenai pendinding api dan membantu proses pembinaan polisi atau *rules* pendinding api dengan lebih jelas lagi tepat.

3 Asas pendinding api

Perlunya pendinding api, terutamanya bila komputer disambung terus ke Internet semakin meningkat, terutama sekali bagi melindungi komputer mahupun jaringan dari serangan cecacing (*worm*) atau dari dicerobohi oleh penceroboh yang tidak bertanggungjawab (*crackers, script-kiddies*).

Sebelum membina atau menggunakan pendinding api, mari kita fahami terlebih dahulu apa fungsinya pendinding api ini. Dari namanya, fungsi utamanya adalah untuk menghalang sambungan yang tidak diingini ke sesuatu komputer (*host*) atau jaringan persendirian (LAN). Ya, pendinding api tidak semestinya diletakkan hanya pada gerbang (*gateway*) atau *router* sahaja, ianya juga boleh dipasang pada hos atau komputer yang dirasakan perlu.

Setiap paket yang melalui module pendinding api akan diperiksa satu persatu dengan SETIAP barisan polisi bermula dari polisi pertama, kedua dan seterusnya. Kalau paket tersebut memenuhi kehendak mana-mana polisi, maka paket itu akan diproses selanjutnya. Umumnya, proses selanjutnya ini bermakna paket ini akan dibuang atau diterima untuk meneruskan perjalanannya. Sekiranya paket tersebut tidak memenuhi sebarang polisi dalam rerantai tersebut, maka, *default policy* akan menjadi penentu. Biasanya *default policy* bagi sesebuah pendinding apai yang baik adalah membuang mana-mana paket sebegini.

4 Strategi pembinaan pendinding api

Seperkara yang mesti diambil perhatian semasa membina atau menambah polisi kedalam rerantai, adalah, paket akan diperiksa dengan polisi yang pertama, kedua dan seterusnya menurut **urutan**. Sebagai contoh; secara analogi, katakan pendinding api adalah seoarang pengawal keselamatan yang mengawal sebuah stadium untuk acara tertentu. Pengawal tersebut mempunyai senarai nombor siri tiket yang dibenarkan masuk kedalam. Katakan senarainya sebegini

1. kesemua tiket yang bernombor dari 0000 hingga 5000 boleh masuk

2. tiket bernombor 4444 tidak dibenarkan masuk
3. yang lain tidak dibenarkan masuk

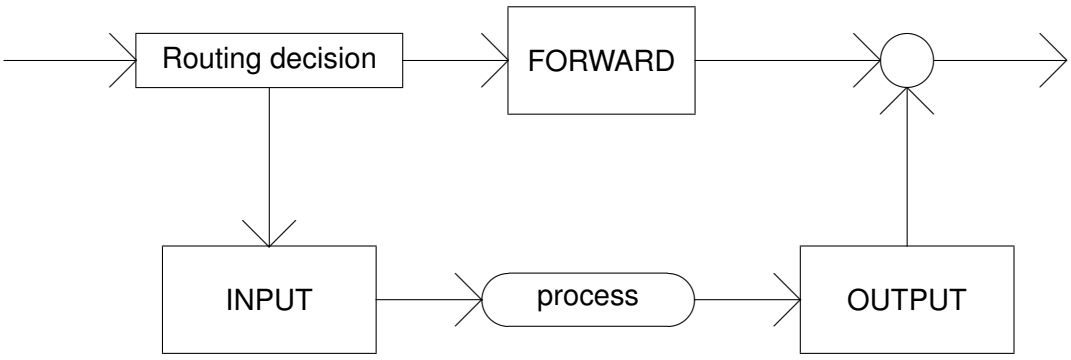
Soalannya, kalau saya memegang tiket bernombor 4444, adakalah saya boleh masuk ke stadium tersebut? Jawapannya, sudah semestinya, saya boleh memasuki ke stadium tersebut, kerana, nombor tiket saya **TELAH MEMENUHI** syarat polisi pada barisan pertama, yakni 4444 adalah nombor yang berada diantara 0000 dan 5000 . Untuk membetulkan keadaan, senarai polisi sepatutnya sebegini.

1. tiket bernombor 4444 tidak dibenarkan masuk
2. kesemua tiket yang bernombor dari 0000 hingga 5000 boleh masuk
3. yang lain tidak dibenarkan masuk

Susunan barisan polisi adalah sangat penting, kerana tersalah letak, apendinding api tidak akan berfungsi seperti apa yang kita hajatkan. Disini kata kunci adalah **SUSUNAN POLISI**.

Setelah mengetahui bagaimana pendinding api memproses setiap paket sambungan, mari kita cuba memahami cara memasukkan polisi-polisi kedalam modul *netfilter*; pendinding api kernel Linux.

5 *netfilter*: pendinding api Linux



Gambarajah ini menunjukkan bagaimana paket yang masuk ke dalam GNU/Linux. Paket akan masuk ke blok *Routing Decision* bagi memastikan paket tersebut masuk ke rerantai yang betul. Kaedah asas yang digunakan adalah dengan melihat **DST** dalam paket. Sekiranya **DST** tersebut bersamaan dengan IP pada sistem tersebut, maka paket itu akan dihantar ke blok rerantai **INPUT**, dan yang lainnya akan di hantar ke blok rerantai **FORWARD**.

Saya andaikan sistem anda sudah berupaya menggunakan modul *netfilter*. Sekiranya anda ada masalah, anda bolehlah merujuk kepada HOWTO tertentu di laman *http://www.tldp.org* dan *http://www.netfilter.org*. Setiap polisi yang kita tambah kedalam kernel akan kekal didalam memori, dan akan terpadam sekiranya kita mengubah atau memadamkannya dengan arahan *iptables* atau mematikan komputer. Sintaks asas iptables adalah :
$\mathrm{iptables}$ -t *table arahan rerantai spesifikasi_polisi opsyen_tambahan*

table adalah jenis fungsi rerantai. *netfilter* mempunyai beberapa *table*, iaitu : **filter, nat, mangle**. Dalam artikel ini, saya tekankan *table* dari jenis *filter* iaitu fungsi asas bagi setiap pendinding api. Sekiranya tiada jenis *table* diberikan pada arahan *iptables* maka, *table* dari jenis *filter* ini akan digunakan.

5.1 *filter table*

Pada *table* jenis *filter*, terdapat 3 rerantai utama, iaitu, **INPUT, FORWARD, OUTPUT**.

5.1.1 INPUT

$$\boxed{\text{SRC}} \longrightarrow \boxed{\text{DST}}$$

Rerantai yang menerima paket yang ingin **masuki** ke sistem itu sendiri sebelum ianya diproses selanjutnya oleh sistem (**DST**).

Contoh, katakan saya telah memasang proses **sshd** pada *port* 22 dan **httpd** pada *port* 80 di hos pelayan **DST**. Katakan saya inginkan benarkan sambungan HANYA ke *port* 22 pada hos ini. Kita perlu meletakkan polisi menyatakan hanya sambungan ke *port* 22 boleh diterima-proses pada rerantai **INPUT** ini. Dengan kata lain, polisi yang harus digunakan adalah membenarkan paket memasuki *DST* pada port 22 (**DPT**) . Kenapa **DST**? Sebab, pelayan tadi adalah destinasi paket yang dimulakan oleh hos pengguna.

$$\boxed{\text{SRC}} \xrightarrow[diterima]{DPT=22} \boxed{\text{DST}}$$

5.1.2 FORWARD

$$\boxed{\text{SRC}} \rightleftarrows \boxed{\text{GW}} \rightleftarrows \boxed{\text{DST}}$$

Rerantai yang menerima paket yang ingin **tumpang lalu** melalui sistem itu sebelum ianya dapat dihantar ke jaringan atau subnet jaringan lain. Gerbang (*GW*) adalah orang tengah bagi kedua-dua hos, **SRC** dengan **DST**, dimana paket-paket kena melaluinya untuk sampai ke destinasinya.

Rerantai ini berguna sekiranya kita ingin menapis atau menghalang sambungan dari pengguna dari dalam LAN kita ke destinasi yang kita tidak ingini, dan begitulah sebaliknya. Sebenarnya cara penggunaan rerantai **FORWARD** adalah mirip dengan rerantai **INPUT**. Yang membezakannya hanyalah alamat **DST** sahaja. Jikalau alamat **DST** sama dengan alamat hos, maka paket akan melalui rerantai **INPUT**. Sebaliknya, jikalau alamat **DST** tidak serupa dengan alamat hos, maka paket itu masuk ke rerantai **FORWARD**.

Diharapkan tips mudah ini dapat membantu memahami konsep pendinding api dengan lebih baik lagi.

5.1.3 OUTPUT

Rerantai **OUTPUT** adalah dimana paket yang **KELUAR** dari sistem, sama ada, paket tersebut merupakan paket yang dibina atau diproses oleh proses-proses sistem. Contoh:

```
# ping google.com
```

Paket ICMP yang berkaitan dengan *ping* dihasilkan oleh proses *ping* dan akan keluar dari sistem melalui rerantai ini. Biasanya, rerantai ini dibiarkan mengeluarkan apa-apa saja paket keluar dari sistem. Sekiranya anda mempunyai matlamat tertentu untuk mengawal apa yang boleh keluar dari sistem, barulah anda perlu merangka bagaimana untuk menapis pada rerantai ini. Bagi memudahkan kefahaman, saya tidak menyentuh secara mendalam mengenai rerantai ini.

5.2 Arahan

Arahan iptables yang asas terdiri dari dibawah

-P Polisi *default* untuk *rerantai*

-F memadam kesesmua polisi pada *rerantai*

-A menambah polisi pada barisan terakhir : *Append*

-D memadamkan polisi pada barisan tertentu : *Delete*

-I meletak pada barisan tertentu : *Insert*

-R menggantikan polisi pada barisan tertentu : *Replace*

-vL menyenaraikan polisi-polisi dalam rerantai : *List*

Walaupun ada beberapa lagi suis opsyen, dalam artikel ini, suis opsyen diatas adalah yang suis yang penting dan kerap digunakan. Saya akan cuba huraikan satu persatu kegunaannya.

5.2.1 [-P] : *default policy*

Andaikata tiada polisi didalam rerantai **INPUT**. Katakan kita ingin menghalang KESEMUA paket dari memasuki hos kita. Arahan dibawah akan mengabaikan segala jenis paket ke sistem. Ini termasuklah ping ke *localhost* itu juga.

```
# iptables -P INPUT DROP
```

Mari kita senaraikan dan lihat bagaimana statusnya.

```
# iptables -nvL
Chain INPUT (policy DROP 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain OUTPUT (policy ACCEPT 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source
destination
```

Anda boleh memfokuskan senarai untuk rerantai tertentu, dengan meletakan nama rerantai dibelakang arahan tadi.

```
# iptables -nvL INPUT
Chain INPUT (policy DROP 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source
destination
```

Pada tahap ini, sila cuba **ping** localhost atau mana-mana alamat dalam jaringan. Contohnya:

```
# ping google.com
```

Sebenarnya, proses *ping* membina paket ping, lalu keluar melalui rerantai **OUTPUT** untuk meneruskan sambungan ke arah *google.com*. Mari kita imbas kembali kepada §*2. Asas TCP/IP*. Walaupun paket *ping* tadi sudah keluar dan sampai ke destinasi (*google.com*), tetapi, paket sahutan ping (*ping reply*) dari *google.com* tidak akan dapat masuk ke sistem kita kerana paket tersebut telah tersangkut dan diabaikan pada rerantai **INPUT**. Oleh kerana rerantai **INPUT** ini berada dihadapan proses sistem, jadi sistem sentiasa menanti kehadiran paket sahutan. Secara ringkas, dari kaca mata proses sistem kita, ianya hanyalah sambungan TCP/IP sehala sahaja.

Sekiranya kita ingin membenarkan sambungan seperti sediakala, kita perlu mengubah kembali *default policy* kepada **ACCEPT**.

```
# iptables -P INPUT ACCEPT
```

5.2.2 [-F]: *flush policies*

Sintaks : **iptables** -F *rerantai*

Sekiranya anda tida meletakkan mana-mana jenis rerantai, *iptables* akan memadam KESEMUA polisi didalam kesemua rerantai. Ini adalah berguna sekiranya kita ingin, mencuci polisi yang lama sebelum anda memulakan polisi yang baru. Contoh:

```
# iptables -F FORWARD
```

Semua polisi didalam rerantai **FORWARD** dipadamkan.

Harus diingat, arahan ini tidak akan mengubah *default policy* pada rerantai. Sila guna arahan **-P**

5.2.3 [-A] : *append*

Sintaks : `iptables` -A *rerantai spesifikasi polisi*

Arahan **-A** ini adalah antara arahan yang paling kerap digunapakai. Arahan ini akan menambahkan polisi baru kedalam rerantai yang kita mahu, pada barisan yang terakhir. Contoh: Saya akan gabungkan arahan-arahan yang telah kita pelajari.

```
# iptables -P INPUT DROP
# iptables -F INPUT
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Mari kita senaraikannya.

```
# iptables -nvL INPUT
Chain INPUT (policy DROP 35 packets, 2100 bytes)
 pkts bytes target prot opt in  out source    destination
    0     0 ACCEPT tcp  --  *   *   0.0.0.0/0 0.0.0.0/0   tcp dpt:22
```

Sekarang mari kita fahamkan barisan ini terlebih dahulu

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

membawa erti, paket dari protokol jenis **tcp** (`-p tcp`) yang menuju ke port (**DPT**) 22 (ssh) (`--dport 22`) akan diterima (`-j ACCEPT`). Paket lain akan diabaikan. Pada tahap ini anda boleh mengakses hos ini hanya pada port 22 sahaja dari komputer lain. Dari contoh ini, saya telah memperkenalkan beberapa suis baru, seperti `-p, --dport, -j`.

Sekarang saya tambah polisi sebegini pula.

```
# iptables -A INPUT -s 192.168.0.0/24 -p tcp --dport 80 -j ACCEPT
```

Ini membawa erti, menerima (`-j ACCEPT`) paket dari puncanya (**SRC**) adalah dari mana-mana hos dalam jaringan **192.168.0.0/24** (`-s 192.168.0.0/24`)yang protokolnya dari jenis tcp (`-p tcp`) yang masuk pada port (**DPT**) 80 (`--dport 80`). Ini bermakna hanya komputer yang berada dalam jaringan 192.168.0.0/24 akan dapat berhubung ke hos ini, itu pun hanya pada port 80 (http).

Katakan, saya diminta supaya menghalang (`-j DROP`) pengguna dari 192.168.0.59 (`-s 192.168.0.59`) dari membuat sambungan ssh (-p tcp –dport 22) ke host ini. Mungkin anda dengan secara spontan akan menambahkan polisi sebegini.

```
# iptables -A INPUT -s 192.168.0.59 -p tcp --dport 22 -j DROP
# iptables -nvL INPUT
Chain INPUT (policy DROP 348 packets, 20880 bytes)
 pkts bytes target prot opt in  out source         destination
  0   0     ACCEPT tcp  --  *   *   0.0.0.0/0      0.0.0.0/0   tcp dpt:22
  0   0     ACCEPT tcp  --  *   *   192.168.0.0/24 0.0.0.0/0   tcp dpt:80
  0   0     DROP   tcp  --  *   *   192.168.0.59   0.0.0.0/0   tcp dpt:22
```

Masih ingat lagikah pada permasalah yang diutarakn di §*4. Strategi pembinaan pembinaan pendiding api* ?

Sama seperti permasalahan di §4, hos 192.168.0.59 masih boleh menghubungi sistem ini pada port 22, kerana paket berkenaan telah pun diterima pada polisi bernombor 1. Jadi macam mana nak ubah supaya polisi nombor 3 berada diatas polisi nombor 1?

5.2.4 [-D] : *delete policy*

Sintaks : `iptables` -D *rerantai nombor_barisan*

Jadi kita ingin memadam polisi bernombor 3 pada rerantai **INPUT** kita tadi. Terdapat 2 cara, tapi saya terangkan cara yang saya kerap gunakan.

```
# iptables -D INPUT 3
```

Wah ini saja? Ya, ini saja. Polisi bernombor 3 sudah dipadam dan sekarang hanya terdapat 2 polisi sahaja dalam rerantai **INPUT** pada saat ini.

```
# iptables -nvL INPUT
Chain INPUT (policy DROP 348 packets, 20880 bytes)
 pkts bytes target  prot opt in  out source         destination
  0   0     ACCEPT  tcp  --  *   *   0.0.0.0/0      0.0.0.0/0   tcp dpt:22
  0   0     ACCEPT  tcp  --  *   *   192.168.0.0/24 0.0.0.0/0   tcp dpt:80
```

Sekarang macam mana kita letakkan polisi baru di barisan atau tempat yang kita inginkan ?

5.2.5 [-I] : *insert policy*

Sintaks : `iptables` -I *rerantai nombor spesifikasi_polisi*

Sekarang mari kita letak polisi yang kita ingin tambah tadi berada ditangga 1.

```
# iptables -I INPUT 1 -s 192.168.0.59 -p tcp --dport 22 -j DROP
# iptables -nvL INPUT
Chain INPUT (policy DROP 348 packets, 20880 bytes)
 pkts bytes target  prot opt in  out source         destination
  0   0     DROP    tcp  --  *   *   192.168.0.59   0.0.0.0/0   tcp dpt:22
  0   0     ACCEPT  tcp  --  *   *   0.0.0.0/0      0.0.0.0/0   tcp dpt:22
  0   0     ACCEPT  tcp  --  *   *   192.168.0.0/24 0.0.0.0/0   tcp dpt:80
```

Lihat polisi tadi sudah berada ditangga 1 dimana paket akan diproses menurut polisi ini terlebih dahulu sebelum periksa oleh polisi-polisi lain.

Sekarang, saya ingin menghadkan hos yang mengkases ke port 22/tcp kepada puncanya hanya dari 192.168.0.0/24. Jadi saya perlu menggantikan polisi bernombor 2 dengan polisi baru. Kita boleh lakukan dengan cara dibawah.

```
# iptables -D 2
# iptables -I INPUT 2 -s 192.168.0.0/24 -p tcp --dport 22 -j ACCEPT
```

Tapi, kita boleh ringkaskan kepada 1 barisan arahan sahaja. Bagaimana?

5.2.6 [-R] : *replace policy*

Sintaks : `iptables` -R *rerantai nombor spesifikasi_polisi*

Cara penggunaanya sama dengan arahan **-I**, mari kita cuba.

```
# iptables -R INPUT 2 -s 192.168.0.0/24 -p tcp --dport 22 -j ACCEPT
# iptables -nvL INPUT
Chain INPUT (policy DROP 348 packets, 20880 bytes)
 pkts bytes target  prot opt in  out source         destination
  0   0     DROP    tcp  --  *   *   192.168.0.59   0.0.0.0/0   tcp dpt:22
  0   0     ACCEPT  tcp  --  *   *   192.168.0.0/24 0.0.0.0/0   tcp dpt:22
  0   0     ACCEPT  tcp  --  *   *   192.168.0.0/24 0.0.0.0/0   tcp dpt:80
```

6 *Statefull Inspection Firewall*

Antara ciri-ciri *netfilter*, adalah ianyalah adalah modul pendinding api yang boleh dikategorikan sebagai *statefull inspection firewall*. Apa bendanya tu? Mari kita semak kembali ke §*2. Asas TCP/IP* dan permasalah *ping* pada §*5.2.1*. Katakan *default policy* untuk rerantai **INPUT** adalah **DROP**. Sistem tidak dapat menerima sahutan *ping* kerana paket tersebut yang kembali ke sistem telah diabaikan sebelum sempat diproses. Jadi apa perlu dibuat?

Untuk menyelesaikan masalah ini pada pendinding api generasi sebelum ini, anda terpaksa buka satu port yang lain bagi paket tersebut dapat masuk dengan memeriksa port punca **SPT**. Contoh untuk ssh dengan simulasi *iptables*.

```
# iptables -P INPUT DROP
# iptables -F
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# iptables -A INPUT -p tcp --sport 22 -j ACCEPT
```

Kenapa jadi sebegini? Katakan sistem ini adalah hos B. Dari hos B, ini saya ingin *ssh* ke hos A

$$B : SPT_B \longrightarrow A : 22$$

Sekarang A kan akan membalas permintaan ssh kembali ke B.

$$A : 22 \longrightarrow B : (DPT = SPT_B)$$

Oleh kerana pada umumnya, A akan membalas permintaan dengan membuka sambungan baru, dengan berhubung kepada port DPT

di B, yang lazimnya sama nilainya dengan nombor port semasa B memulakan sambungan ke A. Oleh kerana SPT_B dipilih secara rawak oleh sistem, maka kita tidak boleh menentukan pada port mana paket yang kembali akan gunakan. Hanya satu jalan saja yang tinggal, iaitu memeriksa port pengirim SPT dan membenarkanya masuk. Polisi kedua pada contoh kali ini dapat merealisasikan kaedah ini.

Ini adalah cara yang terpaksa dilakukan dengan pendinding api generasi lepas seperti *ipchains*. Sekali pandang nampak OK saja. Tapi, sebenarnya ianya membuka pintu risiko yang tinggi, kerana saya boleh menyambung ke sistem tersebut, pada mana-mana port tcp selagi, port puncanya **SPT** adalah 22. Jadi katakan saya dari A ingin menceroboh port 80 di B. Terdapat utiliti yang dapat membina/mengubah paket dengan SPT yang kita kehendaki secara bebas.

$A : 22 \longrightarrow B : 80$ (LEPAS !!!)

Jadi sinilah gunanya *statefull inspection*. Dimana kita tidak perlu lagi membuka port SPT lagi. *statefull inspection* akan merekodkan dan mengesan sambungan yang di**mula**kan oleh sistem ini dan akan membuka port yang berkaitan saja, secara **otomatik** untuk menerima sambungan balas. Pendinding api akan menutup kembali port yang dibuka tadi setelah sessi tamat, secara otomatik juga. Ini akan meningkatkan lagi ciri-ciri keselamatan kerana teknik pencerobohan yang saya terangkan sebelum ini tidak boleh dipakai lagi. Jadi saya sarankan kepada mereka yang masih menggunakan pendinding api tradisional, silalah bertukar ke pendinding api yang mempunyai ciri *statefull inspection* ini.

Sekarang bagaimana untuk menggunakan ciri canggih ini, dengan *netfilter*? Mudah saja, *hafal*kan templet ini saja. Ok, mari kita lihat.

```
# #Templet asas, bagus untuk kegunaan umum. Perisai diri sendiri

# #Cuci rerantai INPUT
# iptables -F INPUT

# #Tutup semua laluan masuk ke sistem
# iptables -P INPUT DROP

# #Benarkan sambungan  dari peranti localhost (loopback device)
# iptables -A INPUT -i lo -j ACCEPT
```

```
# #Mai dah, mai dah, statefull inspections. HAFAL ini pun tak ngapa.
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# #Kalau setakat pengguna dialup/xDSL, ini dah memadai,
# #dan tak payah buka apa-apa port lagi. Abaikan saja barisan selanjutnya.

# #Ok, baru boleh buka port servis kepada pelanggan atau pengguna.
# #Bagus untuk pelayan web, ssh, ...
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
:
:
(dan seterusnya)
```

Macam mana? mudah, kan, untuk membuat PERISAI atau tembok batu keliling sistem kita, disamping kita benar beberapa port tertentu dibuka untuk kegunaan umum. Kalau anda pengguna dial-up atau ASDL, anda tak perlu risau cubaan pencerobohan secara terus melalui jaringan.

7 Polisi untuk gerbang (*gateway*)

Untuk mengawal pergerakkan paket sambungan keluar masuk ke jaringan yang kita selia, pengetahuan *netfilter* ini sangat berguna bagi memastikan penggunaan jaringan dapat dikawal sebaik mungkin. Ianya sama saja dengan pembikinan polisi pada rerantai **INPUT**, cuma anda jangan keliru diantara rerantai **INPUT** dan **FORWARD** sahaja. Berbekalkan templet asas diatas tadi, mari kita bina polisi asas untuk gerbang. Katakan saya ingin mengawal penggunaan dari dalam LAN keluar ke Internet.

$$LAN \longrightarrow eth0 : GW : eth1 \longrightarrow INTERNET$$

Sebelum apa-apa pun mari kita rancang dulu apa yang boleh keluar dan yang patut dihalang. Katakan saya hanya ingin benarkan port-port ini keluar INTERNET.

```
# #Enablekan kernel port forwarding
# echo 1 > /proc/sys/net/ipv4/ip_forward

# #Cuci rerantai FORWARD
# iptables -F FORWARD
```

```
# #Tutup semua laluan masuk ke sistem
# iptables -P FORWARD DROP

# #Ya,  statefull inspections pun boleh dipakai dalam rerantai ini
# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 22 -j ACCEPT
# iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
:
:
(dan seterusnya)
```

Disini kita jumpa lagi suis opsyen baru, `-i eth0` dan `-o eth1`. `-i eth0` adalah memberitahu *netfilter* bahawa paket yang masuk dari peranti (*device*) **eth0** dan paket yang keluar melalui peranti **eth1**. Dengan ini anda tidak perlu meletakkan `-s hostname_or_network`, iaitu dari alamat mana paket itu yang ingin lalu ke Internet. Sekiranya anda menggunakan xDSL ataupun dial-up, anda boleh gantikan `-o eth1` dengan `-o ppp0` atau mana-mana device yang sesuai. Oleh kerana saya tidak merangkumi cara untuk NATkan (*Network Address Translation*) secara mendalam, anda boleh guna tambahkan saja polisi ini pada templet diatas.

```
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

ini sesuai sekiranya anda guna dialup atau mendapatkan IP secara dinamik.

atau sekiranya anda ada talian suwa (*leased line*) dengan IP yang diberikan oleh ISP untuk gerbang anda, sila guna polisi dibawah

```
# iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to IP_AWAM_TETAP_GW
```

8 Kelas Tambahan

Amaran : Saya sarankan anda mengubah polisi pendinding api secara terus pada konsol, bukannya secara jarak jauh. Ada kemungkinan sambungan anda akan terputus, dan anda tidak boleh membuat sambungan lagi, hinggalah anda terpaksa berdepan dengan sistem itu sendiri untuk membetulkannya.

Setelah panjang lebar saya menceritakan mengenai asas TCP/IP dan pendinding api, saya ingin tambah sedikit lagi pada polisi supaya

sistem kita dapat menjawab **ping**. Sila tambah ini pada barisan tertentu yang tertentu supaya sistem dapat menyahut ping.

```
# iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
```

Barisan dibawah ini akan membolehkan pengguna dalam LAN ping kemana-mana host di Internet, tetapi penguna dari Internet tidak boleh ping mana-mana sistem didalam LAN melalui gerbang ini.

```
# iptables -A FORWARD -i eth0 -o eth1 --p icmp --icmp-type 8 -j ACCEPT
```

Anda mungkin mendapat masalah untuk membuat sambungan FTP ke Internet, sekiranya anda membenarkan sambungan ke DPT 21/tcp (`-p tcp --dport 21 -j ACCEPT`), jadi jangan lupa masukkan modul kernel ini terlebih dahulu, atau sila gunakan fungsi **PASSIVE-FTP** yang biasanya terdapat pada klien ftp.

```
# modprobe ip_nat_ftp
```

Oleh kerana setiap polisi disimpan didalam memori, maka ianya akan hilang sekiranya komputer dimatikan atau direboot. Untuk menyimpan polisi yang berada didalam memori, anda boleh menggunakan arahan `iptables-save` dan salurkannya kepada sebuah fail. Untuk meletakkan kembali polisi dari fail tersebut kedalam memori, sila gunakan arahan `iptables-restore` . Dengan ini anda tidak memerlukan lagi arahan dimasukkan satu-persatu dari konsol. Anda juga simpan segala arahan *iptables* kedalam sebuah fail skrip, dan larikan skrip berkenaan bila diperlukan.

Contoh menyimpan dan membaca polisi dari fail.

```
# iptables-save > polisi.netfilter
# iptables-restore < polisi.netfilter
```

Pengguna Red Hat Linux bolehlah menggunakan arahan
`service iptables save`
untuk menyimpan kedalam konfigurasi sistem, supaya x polisi yang disimpan dapat digunakan sebaik sahaja anda memulakan komputer anda.

9 Penutup

Artikel ini menengahkan asas pembinaan polisi untuk pendinding api. Ada banyak lagi fungsi yang canggih dalam *netfilter* ini yang boleh digunakan. Sila rujuk kepada artikel lain bagi memenuhi kehendak dan mempertingkatkan lagi ilmu pengetahuan anda.

- `man iptables`
- *http://www.netfilter.org*
- *http://www.tldp.org*

Perkara kunci (*key-point*) yang harus diingati dan difahami adalah seperti berikut:

- Urutan polisi adalah penting. Rancangkanlah dengan teliti.
- **DST** menentukan paket melalui rerantai **INPUT** ataupun **FORWARD**
  `if (DST==local_ip) then pass to INPUT else FORWARD`
- *statefull inspection* dapat membantu meningkatkan ciri sekuriti

Artikel ini dihasilkan dengan perisian XEmacs, xtexcad dan LaTeX diatas pelantar GNU/Linux dalam masa 24 jam. Sudah tentu terdapat kesilapan dan kesalahan fakta. Sekiranya terdapat sebarang cadangan dan pembetulan, jangan malu-malu untuk berhubung dengan saya melalui email.

Wassalamz, terima kasih.
– *Ilmu itu ilmu Allah*