25 April 2008

PANDUAN PRAKTIS FIREWALL DENGAN IPTABLES

Pendahuluan

Saat ini rasanya hampir tidak ada komputer yang tidak tersambung ke jaringan/internet, baik itu melalui modem dialup, modem ADSL, maupun dedicated LAN. Hal ini menunjukkan pengaruh globalization di dalam beberapa dekade ini, dimana salah satu imbasnya adalah masuknya kita ke dalam 'internet society' - sebuah komunitas global yang terhubung melalui internet.

Hal ini tentu saja memberikan banyak pengaruh positif seperti makin cepatnya tersebarnya informasi dari satu negara ke negara lain, makin lancarnya transaksi bisnis lintas negara, dll. Dari sisi mikro, kini telah merupakan hal yang umum bila perusahaan berlangganan koneksi internet dedicated. Dengan berlangganan internet dedicated, maka selain lebih ekonomis, perusahaan pun dapat lebih leluasa dan cepat di dalam mengakses informasi/kesempatan bisnis lewat internet. Namun hal ini juga memberikan dampak negatif, atau mungkin lebih tepat kalau kita memakai istilah 'tantangan'.

Mengapa tantangan? Sebab dibandingkan sewaktu munculnya era internet di tahun 1990an, kini internet telah menjadi lingkungan yang sangat kejam. Tidak perduli apakah kita admin yang baru belajar ataupun yang sudah pengalaman, kita harus selalu waspada. Waspada dalam hal apa? Di dalam konteks ini adalah waspada di dalam mengatur siapa/apa yang boleh/tidak boleh mengakses masuk/keluar network kita. Bisa dibayangkan apa yang akan terjadi bila ada cracker yang berhasil membobol server kita. Mulai dari deface website, abuse bandwidth, sampai penghapusan data. Ini berarti kita harus selalu menjaga keamanan network dan server2 kita.

Ada banyak cara yang harus dilakukan di dalam menjaga keamanan network dan server, namun di tulisan kali ini kita akan konsentrasi membahas dari sisi firewall. Apa itu firewall? Seperti arti harafiah katanya, yaitu: 'tembok api'. Di firewall ini kita melakukan proses penyaringan trafik network apa dan bagaimana yang kita perbolehkan/larang. Di dalam konsep networking, semua service networking (seperti web, ftp, mail, dns, dll) berjalan melalui jalur2 yang kita namakan 'port'. Masing2 service tersebut memiliki jalurnya sendiri, yaitu port2 dengan nomor tersendiri, seperti service:
- web ---> port tcp 80, 445
- ftp ---> port tcp 20, 21
- mail ---> port tcp 25, 110
- dns ---> port tcp/udp 53
- Dll, lihat daftar lengkapnya di file /etc/services

Dua pendekatan setup firewall

Di mailing list sering ada yang bertanya: "Port apa saja sih yang harus kita TUTUP?" Ini adalah pendekatan '*negative list*', dimana secara DEFAULT semua port kita BUKA, baru kemudian satu per satu kita TUTUP port yang diinginkan.

Pendekatan kedua adalah: '*positive list*', yaitu dimana secara DEFAULT semua port di TUTUP, dan baru kemudian satu per satu kita BUKA port yang diinginkan.

Mana dari kedua pendekatan itu yang sebaiknya kita pakai? Berikut ini adalah summarynya:

1. Negative List:
- **Mudah disetup**
- **Beresiko lupa menutup port**

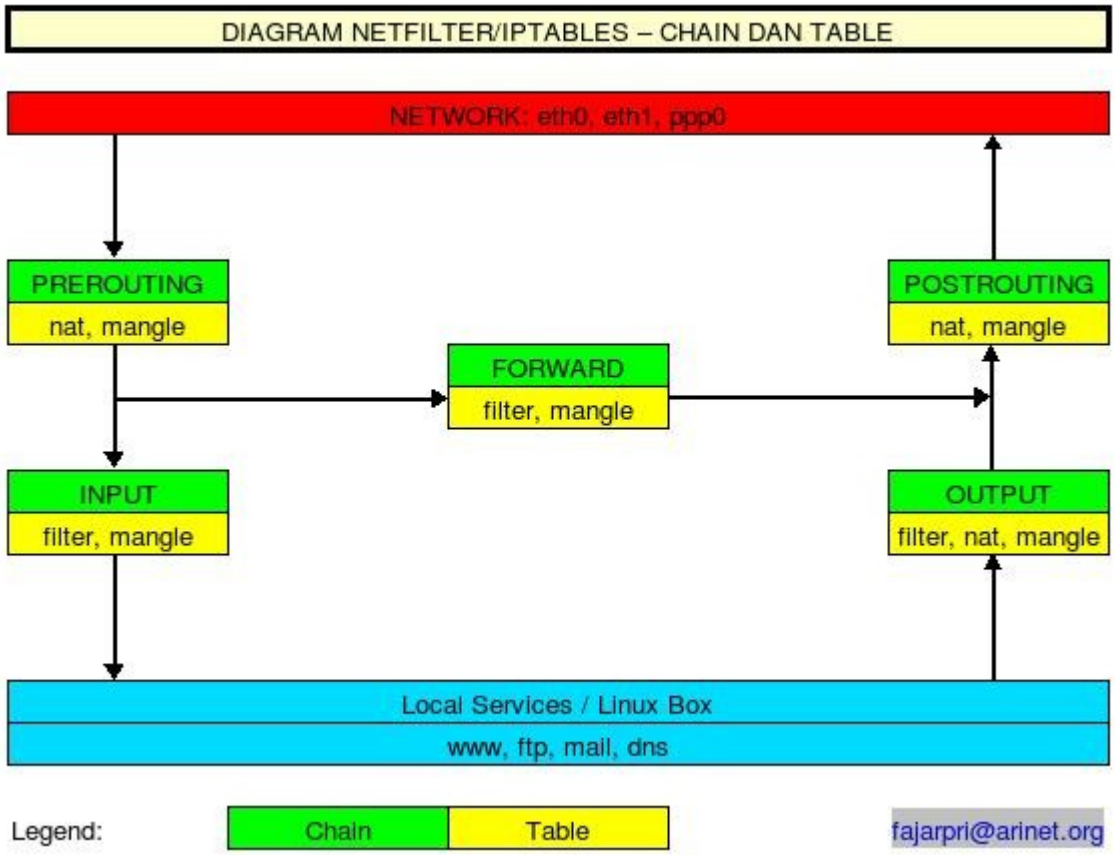
2. Positive List:
- **Sangat secure, sebab semua port tertutup**
- **Relatif lebih sulit disetup**

Sampai sejauh ini semoga konsep dasar firewall telah cukup jelas. Sekarang bagaimana sebenarnya kita mengatur firewall di Linux? Untuk ini kita menggunakan yang namanya *Netfilter*. Netfilter adalah framework yang menyediakan cara untuk memanipulasi packet2 network lewat kernel Linux. Sedangkan *iptables* adalah utility yang digunakan untuk mengatur hal tersebut. Ooohh... gitu :)

Konsep Netfilter/Iptables

Sebenarnya bagaimana sih cara bekerja iptables sebagai firewall? Seperti dapat kita lihat di gambar 1, di tengah2 bawah adalah server Linux kita. Di atas adalah Network, baik local maupun internet. Di gambar itu juga terlihat ada 5 buah kotak berwarna hijau kuning. Kotak-kotak ini melambangkan titik/lokasi tempat kita bisa melakukan filtering maupun manipulasi terhadap paket network yang sedang lewat.

[Gambar 1, Diagram Netfilter]



Berikut ini adalah definisi dari masing2 titik tersebut:
1. **PREROUTING**, titik dimana kita bisa memanipulasi paket network sebelum dia memasuki keputusan routing, apakah ia akan masuk ke dalam Linux kita atau cuma sekedar 'lewat'.
2. **INPUT**, titik dimana kita bisa melakukan pemeriksaan terhadap paket network yang akan MASUK ke dalam Linux kita.
3. **OUTPUT**, titik dimana kita melakukan pemeriksaan terhadap paket network yang dihasilkan oleh Linux kita KELUAR sebelum routing.
4. **FORWARD**, titik dimana kita melakukan pemeriksaan terhadap paket network yang cuma numpang LEWAT Linux kita.
5. **POSTROUTING**, titik dimana kita bisa melakukan manipulasi terhadap paket yang akan keluar dari Linux kita.

Kelima 'titik' di atas kita sebut **CHAIN** (rantai) di dalam syntax Iptables.

Lalu bagaimana dengan kotak yang berwarna kuning? Ini disebut table. Table adalah tempat rule2/aturan2 yang kita buat disimpan. Ada 3 buah table, secara general dapat kita definisikan:
1. **Table filter**, adalah tempat rule2 yang berkaitan dengan boleh/tidaknya suatu paket network melewati sebuah CHAIN di atas.
2. **Table nat**, adalah singkatan dari Network Address Translation, yaitu table tempat rule2 yang berkaitan dengan manipulasi suatu paket network ketika melewati CHAIN PREROUTING, POSTROUTING, dan OUTPUT.
3. **Table mangle**, adalah tempat rule2 yang berkaitan dengan manipulasi suatu paket network untuk keperluan advance, seperti QOS (quality of service), packet marking, dll.

Syntax Iptables
Iptables memberikan keleluasaan yang sangat besar kepada kita di dalam mengatur firewall. Kita bisa membuat rule mulai dari yang general/umum sampai yang detail sekali.

Syntax Iptables seperti ini (perhatikan huruf besar kecil berpengaruh):

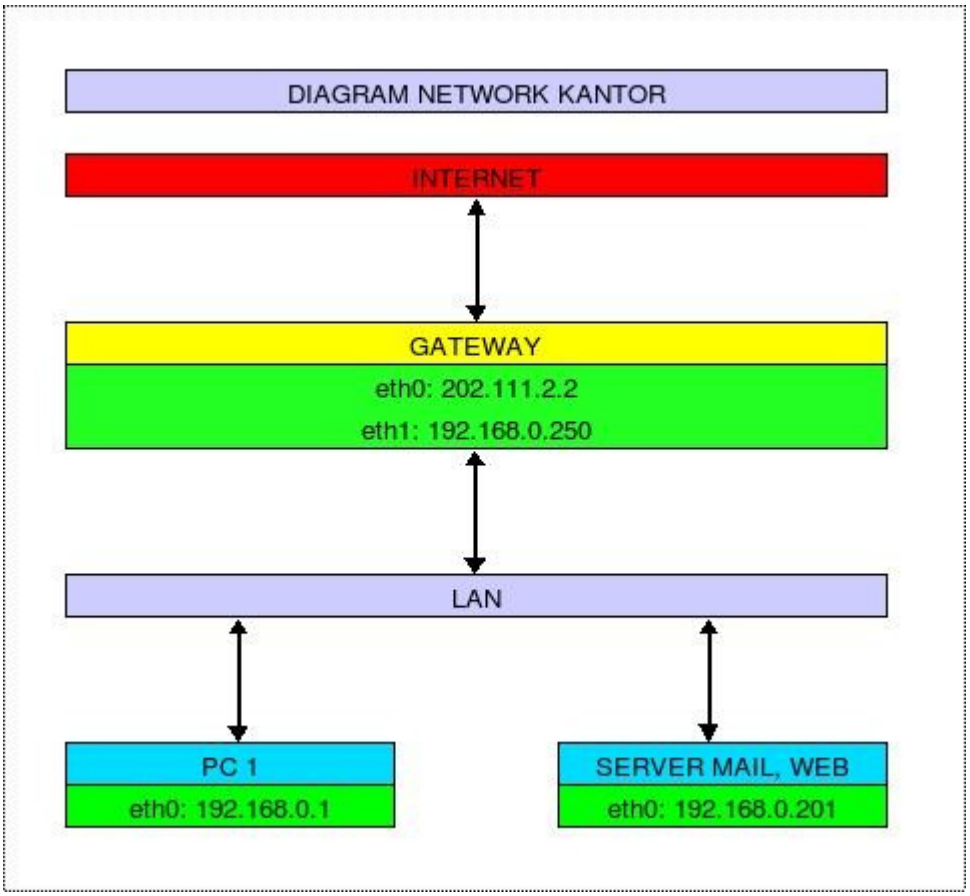
```
iptables [-t table] -[AD] CHAIN rule-specification [options]
iptables [-t table] -I CHAIN [rulenum] rule-specification [options]
iptables [-t table] -R CHAIN rulenum rule-specification [options]
iptables [-t table] -D CHAIN rulenum [options]
iptables [-t table] -[LFZ] [CHAIN] [options]
iptables [-t table] -P CHAIN target [options]
```

Keterangan:
`-I` : insert, bila menggunakan I, maka secara default rule akan menempati baris nomor 1. Kita bisa menyebutkan ingin insert di baris keberapa.
`-A` : append, rule akan ditambahkan di baris paling bawah.
`-D` : delete, menghapus rule baris ke-berapa.
`-R` : replace, mengganti rule baris ke-berapa.
`-L` : list, menampilkan rule2 yang ada.
`-P` : policy, mengubah policy suatu chain.

Wah cukup rumit yah. Memang mungkin sebaiknya kita membahas contoh penggunaan iptables menggunakan cerita. Kita gunakan contoh gambar 2. Disini ceritanya kita berperan sebagai seorang admin Linux di kantor. Kantor kita berlangganan internet dedicated dan kita diminta melakukan:
1. Mengamankan gateway dari kemungkinan serangan baik dari luar maupun dalam.
2. Secara default user tidak dapat melakukan koneksi ke gateway dan internet.
3. Sharing internet.
4. Membuat transparent proxy di gateway.
5. Mengatur agar kita bisa membuat sebuah web dan mail server yang

terlindungi di belakang gateway.

[Gambar 2, Diagram Network Kantor]



Wihhh... cukup banyak nih. Apalagi kita akan menggunakan pendekatan 'positive list', dimana secara default kita akan menutup seluruh port yang ada. Sebelum kita mulai, artikel ini mengasumsikan kita menggunakan distro Centos 5, tapi distro lain pun kurang lebih akan sama step by stepnya. Oya, juga sangat disarankan agar melakukan setting ini langsung di depan layar server Linux kita. Boleh saja sih mengatur iptables secara network, tapi pastikan bahwa langkah pertama yang kita lakukan adalah membuat sebuah rule yang membolehkan akses dari PC kita.

Kita mulai yah.
Tugas 1. Mengamankan gateway.
Di Centos paket iptables secara default sudah terinstall dan aktif, tapi masih dalam kondisi kosong rule2nya.

*** Bila karena suatu hal kita ingin menghapus semua rule yang telah kita buat, maka commandnya:

```
iptables -F
```

*** Bila ingin menghapus sebuah rule saja:

```
iptables -nL --line-numbers
iptables -D INPUT|OUTPUT|FORWARD [barisnya]
```

, misalnya:

```
iptables -D INPUT 3
```

Sangat disarankan agar kamu melakukan command2 iptables ini dari mode text demi kemudahan. Sebab bila dilakukan dari mode GUI, ada kemungkinan Xwindow kita hang, kita harus membuat rulenya dahulu agar tidak hang. Tapi agar menjadi simple di tahap awal ini, maka disarankan kita melakukannya dari mode text dahulu yah (ketik init 3, atau Ctl+Alt F1).

Untuk melihat status iptables:

```
[root@server ~]# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Penjelasan:
Chain INPUT (policy ACCEPT) = Chain INPUT memiliki policy ACCEPT, jadi secara default paket network boleh MASUK ke server Linux kita. Demikian juga dengan chain lainnya, masih diperbolehkan (ACCEPT).

Ok, kita akan segera mengamankan server Linux gateway kita. Disini karena saya tidak sedang berada di depan server Linux saya, maka saya mengaturnya melalui network. Oleh karena itu hal pertama yang harus saya lakukan adalah membuat sebuah rule yang MEMBOLEHKAN akses ssh dari notebook saya ke server Linux tersebut. Hal ini harus dilakukan paling pertama, sebab kalau tidak saya akan 'terblok' sendiri oleh server Linux saya.

```
iptables -I INPUT -s 192.168.0.227 -j ACCEPT
```

Arti command di atas adalah:
Me - INSERT sebuah rule di chain INPUT, dimana semua trafik yang bersumber (-s) (SOURCE) dari IP 192.168.0.227, di -j (JUMP) ACCEPT (boleh). -j atau JUMP ini adalah action dari sebuah rule, apakah ingin kita bolehkan/reject/drop dll. Ok jelas yah?

Sekarang kita lihat lagi kondisi iptables kita:

```
[root@server ~]# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT      all  --  192.168.0.227          0.0.0.0/0       <--- perhatikan
muncul rule ini

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Terlihat bahwa di chain INPUT sekarang ada baris rule yang menyatakan target: ACCEPT, prot (protocol): all, opt (option): kosong, source: 192.168.0.227, destination: 0.0.0.0/0 (all). Ok, sampai disini mulai ada gambaran yah?

Lanjutnya, bagaimana cara mengamankan gateway Linux kita ini? Bila menggunakan pendekatan 'negative list', maka kita harus membuat puluhan/ratusan rule untuk melakukan penutupan port2 yang ada. Hal ini tentu cukup merepotkan. Oleh karena itu kita menggunakan pendekatan 'positive list', dimana kita mengubah POLICY CHAIN yang ada dari ACCEPT menjadi DROP. Oya sebelumnya kita jelaskan, policy ini bekerjanya adalah sebagai 'default action' apa yang harus dilakukan terhadap suatu paket bila tidak ada rule yang cocok untuk paket itu. Dalam kondisi awal iptables kita ini sekarang adalah: ACCEPT. Jadi semua paket bebas keluar masuk, lewat gateway Linux kita.

Untuk mengamankan gateway Linux kita, ubah policy chain2 yang ada menjadi drop.

```
iptables -P INPUT DROP
```

Coba kita list lagi seperti apa kondisinya iptables kita:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)       <--- perhatikan berubah policynya jadi drop
target     prot opt source               destination
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Ok sekarang sudah terlihat bahwa policy chain INPUT telah menjadi DROP. Ini artinya apa? Secara default semua paket network TIDAK diperbolehkan MASUK ke dalam server gateway Linux kita. Nah sudah cukup secure kan? Sekarang marilah kita mengubah

policy chain yang 2nya lagi menjadi DROP juga. Eh, tapi nanti dulu. Karena saat ini posisi saya adalah di notebook, yang melakukan koneksi ssh ke server, maka kita harus buat dahulu sebuah rule untuk notebook saya. Lho, bukannya tadi sudah? Iya tadi yang sudah adalah sebuah rule yang MEMBOLEHKAN akses INPUT dari notebook ke server, bagaimana dengan rule yang MEMBOLEHKAN akses OUTPUT dari server ke notebook? Ini juga mesti kita definisikan, sebab kalau tidak koneksi ssh dari notebook ke server sih memang bisa, tapi trafik balasannya dari server ke notebook akan terblok.

Oleh karena itu kita perbolehkan dahulu trafik network yang KELUAR dari server ke DESTINATION IP notebook saya:

```
iptables -I OUTPUT -d 192.168.0.227 -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT      all  --  0.0.0.0/0            192.168.0.227   <--- perhatikan
muncul rule ini.
```

Baru kemudian aman kita mengubah policy untuk chain OUTPUT dan FORWARD menjadi DROP:

```
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy DROP)      <--- perhatikan berubah policynya menjadi
DROP.
target     prot opt source               destination

Chain OUTPUT (policy DROP)      <--- ini juga.
```

```
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            192.168.0.227
```

Catatan: sebenarnya kita bisa menggunakan feature 'connection tracking' dari iptables agar kita tidak perlu membuat rule untuk koneksi OUTPUT tadi. Tapi ini nanti kita bahas.

Oya, secara tidak sadar kita ternyata telah melakukan **Tugas no. 2** juga, yaitu secara default user tidak dapat melakukan koneksi ke internet dan gateway :) Jadi user yang ada di PC 1 seperti di gambar, dia tidak akan dapat melakukan koneksi ke internet dan gateway dalam bentuk apapun, ssh, ftp, web, dll.

Tugas 3. Sharing Internet.
Nah tentu dengan memiliki server gateway Linux yang secure tidak akan berguna banyak kalau ia tidak dapat berfungsi seperti yang diharapkan, yaitu dalam hal ini menyediakan akses internet untuk user kantor kita.

Untuk sharing internet:
Pertama yakinkan bahwa setup routing di gateway kita sudah benar.
Kedua, aktifkan ip forwarding di `/etc/sysctl.conf`:

```
net.ipv4.ip_forward = 1
```

Ketiga baru setup iptablesnya:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Hasilnya:

```
[root@server ~]# iptables -t nat -nL    <--- perhatikan ada -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0

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

Keterangan:
Kita membuat sebuah rule di table nat dan di chain POSTROUTING, dimana semua paket network yang keluar dari eth0 kita akan di MASQUERADE. Apakah ini? Ini artinya adalah semua paket network yang berasal dari IP LAN kita (192.168.0.x) akan "dibungkus" atau di masquerade dengan IP Public gateway kita. Mengapa perlu dibungkus dengan IP Public? Sebab IP LAN (kelas C) tidak diperbolehkan di routing keluar ke internet.

Terus, kenapa ip forwarding perlu dihidupkan pula? Sebab untuk sharing internet, paket

network dari LAN kita akan 'masuk' ke eth1 kita, kemudian 'dilempar' keluar ke internet melalui eth0 kita. Jadi fungsi ip forwarding di kernel kita perlu dihidupkan di `/etc/sysctl.conf` itu. Lihat gambar 2.

Tapi hal ini saja tidak cukup, sebab dari sisi iptables kita perlu mengatur pula siapa2 saja atau apa2 saja yang kita perbolehkan melewati chain FORWARD dari iptables kita. Karena policy chain FORWARD tadi kita setel menjadi DROP, maka kita harus mengatur siapa saja yang boleh lewat.

Misalkan kita ingin membolehkan PC 1 untuk browsing internet, maka kita buat rule:

```
iptables -A FORWARD -p tcp --dport 80 -s 192.168.0.1 -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:80

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            192.168.0.227
```

Keterangan:
Kita membuat sebuah rule di chain FORWARD, dimana paket network yang berupa protocol tcp dan di port 80(www), yang berasal dari IP 192.168.0.1, diperbolehkan.

Coba browsing dari PC 1, uupppss ternyata blom bisa. Hal ini ternyata karena trafik DNS untuk meresolve domain name belum diperbolehkan. Disini kita asumsi bahwa settingan DNS di PC 1 adalah memakai DNS di ISP. Oleh karena itu kita buatkan pula rulenya:

```
iptables -I FORWARD -p tcp --dport 53 -j ACCEPT
iptables -I FORWARD -p udp --dport 53 -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination
```

```
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:53
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:80

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            192.168.0.227
```

Ok, sharing internet browsing sudah bisa. Bagaimana dengan chatting? Misalnya Yahoo Messenger? Kita mesti membuat rule spesifik untuk itu juga. Yahoo messenger memakai port tcp 5050, jadi:

```
iptables -A FORWARD -p tcp --dport 5050 -s 192.168.0.1 -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:53
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:80
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:5050

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            192.168.0.227
```

****
Connection Tracking
****

Sebelum kita membahas lebih lanjut, ada baiknya kita sentuh konsep connection tracking. Tadi di atas kita telah melihat bahwa agar koneksi ssh saya dari notebook ke server tidak putus ketika kita mengubah policy chain INPUT dan OUTPUT menjadi DROP, maka kita harus mendefinisikan rule 2 di kedua buah chain itu. Cukup merepotkan yah. Untuk mempermudah hal ini, kita bisa memanfaatkan fungsi connection tracking dari iptables, yaitu dimana ketika kita sudah mendefinisikan sebuah rule di chain tertentu, maka trafik network yang terkait dengan rule tersebut tidak perlu disebutkan lagi. Jadi dalam hal ssh dari notebook ini, kita cukup mendefinisikan rule di chain INPUT saja, yang di OUTPUTnya tidak perlu lagi.

Caranya adalah:

```
iptables -I INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -I OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT       all   --   0.0.0.0/0             0.0.0.0/0           state
RELATED,ESTABLISHED
ACCEPT     all  --  192.168.0.227        0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT       all   --   0.0.0.0/0             0.0.0.0/0           state
RELATED,ESTABLISHED
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:53
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:80
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:5050

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT       all   --   0.0.0.0/0             0.0.0.0/0           state
RELATED,ESTABLISHED
ACCEPT     all  --  0.0.0.0/0            192.168.0.227

**********************
```

Tugas 4. Transparent Proxy.

Apakah itu transparent proxy? Ia adalah kondisi dimana kita bisa "memaksa" semua trafik web yang berasal dari LAN kita agar diarahkan masuk ke dalam squid/proxy kita. Jadi di PC client kita tidak perlu mengatur browser Firefox untuk menggunakan proxy tertentu lagi. Semuanya akan otomatis 'dipaksa' masuk ke dalam squid kita.

Detail cara menyetel squid untuk sebagai transparent proxy bisa dibaca di artikel ini:
http://linux2.arinet.org/index.php?option=com_content&task=view&id=39&Itemid=2

Singkatnya, edit `/etc/squid/squid.conf`:

```
httpd_accel_host virtual
httpd_accel_port 80
```

```
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Untuk squid versi 2.6 keatas, editnya:

```
http_port 3128 transparent
```

Dari sisi iptables, kita lakukan ini:

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-
port 3128
```

Dan kita perbolehkan squid untuk mengakses internet:

```
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 53 -j ACCEPT
iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  0.0.0.0/0             0.0.0.0/0           tcp dpt:80
redir ports 3128

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0

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

Tugas 5. Mengatur agar kita bisa membuat sebuah web dan mail server yang terlindungi di belakang gateway.

Coba kita lihat lagi gambar 2. Di tugas 5 ini, kita harus mengatur agar trafik dari internet yang masuk ke eth0 di port web(tcp 80) dan mail(tcp 25,110) akan di 'forward' ke port yang bersangkutan di IP Mail Server kita (192.168.0.201).

Commandnya untuk IP forwarding ini adalah:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to
192.168.0.201:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j DNAT --to
192.168.0.201:25
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 110 -j DNAT --to
```

```
192.168.0.201:110
```

Hasilnya:

```
[root@server ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 redir ports 3128
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 to:192.168.0.201:80
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:25 to:192.168.0.201:25
DNAT          tcp   --   0.0.0.0/0                0.0.0.0/0               tcp  dpt:110  to:
192.168.0.201:100

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0

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

Kita mesti membuka pula untuk di chain FORWARDnya:

```
iptables -A FORWARD -p tcp -i eth0 --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 --dport 25 -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 --dport 110 -j ACCEPT
```

Sebenarnya apa gunanya membuat Web dan Mail Server di belakang gateway seperti ini? Selain untuk keamanan, guna lainnya adalah agar kita dapat 'memaksa' user LAN kita agar kalau ingin mengirim email, harus menggunakan mail server kita tersebut. Mengapa demikian? Agar user tidak dapat seenaknya mengirim email menggunakan smtp lain di internet, dan juga untuk mencegah program2 spyware dan virus mengirim spam ke internet dari LAN kita.

Ok, trafik web dan email dari internet ke LAN kita sudah diatur, lanjutnya adalah mengatur yang dari Web dan Mail server kita ke internet:

```
iptables -A INPUT -i eth1 -p tcp --dport 80 -s 192.168.0.251 -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --dport 25 -s 192.168.0.251 -j ACCEPT
```

Hasilnya:

```
[root@server ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
ACCEPT     all  --  192.168.0.227        0.0.0.0/0
ACCEPT     tcp  --  192.168.0.251        0.0.0.0/0           tcp dpt:80
ACCEPT     tcp  --  192.168.0.251        0.0.0.0/0           tcp dpt:25

Chain FORWARD (policy DROP)
target     prot opt source               destination
```

```
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:53
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:80
ACCEPT     tcp  --  192.168.0.1          0.0.0.0/0           tcp dpt:5050

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
ACCEPT     all  --  0.0.0.0/0            192.168.0.227
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           tcp:80
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           udp:53
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           tcp:80
```

****

Melihat DETAIL kondisi Iptables kita. Gunakan 'v' dan --line-numbers

****

```
[root@server ~]# iptables -nvL --line-numbers
Chain INPUT (policy DROP 4 packets, 272 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      515 35388 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
2     1302 96800 ACCEPT     all  --  *      *       192.168.0.227        0.0.0.0/0
3        0     0 ACCEPT     tcp  --  eth1   *       192.168.0.251        0.0.0.0/0           tcp dpt:80
4        0     0 ACCEPT     tcp  --  eth1   *       192.168.0.251        0.0.0.0/0           tcp dpt:25

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
2        0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
3        0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:53
4        0     0 ACCEPT     tcp  --  *      *       192.168.0.1          0.0.0.0/0           tcp dpt:80
5        0     0 ACCEPT     tcp  --  *      *       192.168.0.1          0.0.0.0/0           tcp dpt:5050

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      312 37296 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
2      686 74712 ACCEPT     all  --  *      *       0.0.0.0/0            192.168.0.227
3      686 74712 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
4      686 74712 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
5      686 74712 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:53
```

Apa gunanya kita menampilkan nomor baris dari rule2 tersebut? Gunanya adalah untuk memudahkan kita melakukan beberapa command seperti:
Untuk menghapus sebuah rule yang salah/tidak diperlukan lagi:

```
iptables -D INPUT 2
```

Ini artinya menghapus rule di baris no. 2 dari chain INPUT. Lho, bukannya gampang dilihat? Iya kalau rulenya sedikit, kalau rulenya sudah puluhan, pusing juga melihatnya kalau tidak ada nomor barisnya :)

```
iptables -I INPUT 3
```

Ini artinya meng-insert rule di baris 3 di chain INPUT.

****

Save Iptables

Semua rule yang telah kita buat, bisa kita save dengan command:
`service iptables save.`
Dia akan di save di `/etc/sysconfig/iptables`

******
REVIEW
******
Ok cukup melelahkan juga yah? Ada baiknya kita review kembali semua rule yang telah kita buat di atas sambil menambahkan beberapa rule yang berguna.
Berikut ini adalah command2 lengkapnya:

```
#### Normalkan dahulu policy yang ada
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#### Hapus semua rule yang ada (Flush)
iptables -F

#### Siapkan rule untuk notebook kita agar tidak kena blok (sesuaikan IPnya
dengan IP kamu)
iptables -I INPUT -s 192.168.0.227 -j ACCEPT

#### Secara localhost kita mesti membolehkan akses
iptables -I INPUT -i lo -j ACCEPT
iptables -I OUTPUT -o lo -j ACCEPT

#### Rule untuk connection tracking, agar kita mudah membuat rule2
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#### Rule untuk sharing internet
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

#### Rule untuk DNS (asumsi bahwa kita menggunakan DNS ISP)
iptables -A FORWARD -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -p udp --dport 53 -j ACCEPT

#### Rule untuk membolehkan PC 1 (192.168.0.1) bisa browsing internet dan
chatting yahoo messenger
iptables -A FORWARD -p tcp --dport 80 -s 192.168.0.1 -j ACCEPT
```

```
## (Bila nanti kita sudah menggunakan transparent proxy, maka rule ini
tidak perlu lagi)

iptables -A FORWARD -p tcp --dport 5050 -s 192.168.0.1 -j ACCEPT

#### Rule untuk membuat transparent proxy squid
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-
port 3128

#### Rule untuk membolehkan squid dan gateway sendiri mengakses internet
iptables -A INPUT -p tcp --dport 3128 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 53 -j ACCEPT
iptables -A OUTPUT -p udp --dport 53 -j ACCEPT

#### Rule untuk melakukan port forwarding dari eth0(internet) di port tcp
80,25,100 ke mail server kita di 192.168.0.251
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to
192.168.0.251:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j DNAT --to
192.168.0.251:25
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 110 -j DNAT --to
192.168.0.251:110

#### Rule untuk membolehkan akses Web dan Mail dari internet ke Web dan
Mail server kita
iptables -A FORWARD -p tcp -i eth0 --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 --dport 25 -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 --dport 110 -j ACCEPT

#### Rule agar Web dan Mail server kita bisa mengakses internet
iptables -A FORWARD -p tcp --dport 80 -s 192.168.0.251 -j ACCEPT
iptables -A FORWARD -p tcp --dport 25 -s 192.168.0.251 -j ACCEPT

#### Mengaktifkan fungsi logging agar kita mudah troubleshoot
iptables -A INPUT -m limit --limit 2/m --limit-burst 2 -j LOG --log-prefix
'** INPUT DROP ** '
iptables -A FORWARD -m limit --limit 2/m --limit-burst 2 -j LOG --log-
prefix '** FORWARD DROP ** '
iptables -A OUTPUT -m limit --limit 2/m --limit-burst 2 -j LOG --log-prefix
'** OUTPUT DROP ** '

#### Terakhir mengatur default policy menjadi drop
```

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#### Jangan lupa di save
service iptables save

#### SELESAI
```

Rule2 tersebut di atas dapat di download dari http://linux2.arinet.org di menu **Download > Artikel > Linux Admin > Sort Berdasarkan Submit Date > iptables-praktis-script.sh**

Penutup dan kesimpulan
Demikianlah pedoman praktis iptables ini. Fiuuuhh..., capek juga yah :) Tapi memang di tengah kondisi internet yang kejam sekarang ini, memasang firewall untuk melindungi network kita sudah merupakan keharusan.

Sebenarnya ada banyak tersedia tool/utility untuk mengatur iptables ini seperti: *shorewall, firewall-builder*, dll. Tidak ada salahnya kita menggunakan tool2 tersebut, tapi setidaknya dengan telah mengetahui konsep firewall dengan iptables, kita akan dapat lebih mudah mengaturnya entah langsung dari iptablesnya ataupun menggunakan tool2 bantuan lainnya.

Sebagai penutup, penulis sadar panduan ini jauh dari sempurna, dan oleh karenanya, segala saran, kritik, koreksi... Monggo.... :)
Terima kasih banyak sebelumnya atasnya. Selamat belajar.

Sentul City, 26 April 2008
Fajar Priyanto, a linux lover :)