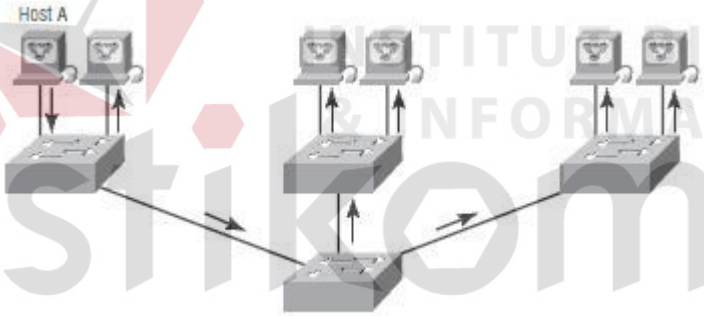
BAB II

LANDASAN TEORI

2.1 VLAN (*Virtual Local Area Network*)

Gambar 2.1 menunjukan suatu jaringan dengan desain *Flat Network.* hal ini berarti, setiap paket *broadcast* yang dikirim, diterima oleh setiap *device* meskipun *device* tersebut tidak membutuhkan paket data. Secara *default*, *Router* memungkinkan pengiriman paket *broadcast* hanya dalam satu jaringan, sedangkan *switch* meneruskan paket *broadcast* ke semua segmen. Pada Gambar 1.1 host A mengirimkan paket *broadcast* dan semua *port* pada semua *switch* mem-*forward paket* tersebut kecuali *port* yang mengirimkan *broadcast*.

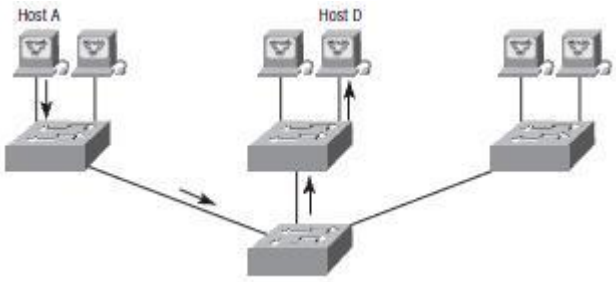


Gambar 2.1 Struktur Jaringan Datar
Sumber : CCNA study guide (hal 553)

Lihat pada Gambar 2.2 gambar tersebut merupakan gambar *switched network* yang menunjukan *Host* A mengirim *frame* kepada *host* D. *frame* dikirim hanya kepada *Host* D. hal ini merupakan suatu perkembangan dari *Hub networks.*

Layer 2 *switched network* menciptakan segmen *collision domain* individu untuk setiap perangkat dihubungkan ke setiap *port* di *switch*. hal ini membebaskan dari kendala jarak *Ethernet*, sehingga jaringan yang lebih besar bisa dibangun.

Namun seringkali, setiap kemajuan baru datang dengan masalah baru. Sebagai contoh, semakin besar jumlah pengguna dan perangkat, semakin banyak lagi paket *broadcast* dan setiap *switch* yang harus ditangani.



Gambar 2.2 Manfaat dari *switched network*
Sumber : CCNA study guide (hal 553)

Masalah lain yang perlu diperhatikan adalah keamanan. karena di dalam *layer* 2 *internetwork switch*, semua *user* dapat mengakses semua perangkat secara *default*. Selain itu pengiriman paket tidak bisa dihentikan, dan tidak bisa menghentikan pengguna dari berusaha untuk menanggapi paket *broadcast*. Hal Ini berarti pilihan keamanan terbatas untuk menempatkan *password* pada *server* dan perangkat lain.

Hal tersebut dapat diatasi dengan membuat *Virtual Local Area Network* (VLAN). kita dapat memecahkan banyak masalah yang terkait dengan *layer* 2 *switching* dengan VLAN.

Berikut adalah bagaimana VLAN memudahkan manajemen jaringan:

1. Memudahkan penambahan jaringan (*segment*), dengan hanya mengkonfigurasi *port* ke VLAN yang sesuai.
2. Pengguna yang memerlukan tingkat keamanan yang tinggi dapat digolongkan kedalam VLAN tertentu sehingga pengguna di luar VLAN tidak dapat berkomunikasi dengan mereka.

3. VLAN mengelompokkan *user* secara logis, VLAN dapat dianggap independen karena tidak terbatas pada lokasi fisik atau geografis.
4. VLAN meningkatkan keamanan jaringan. VLAN memperbanyak jumlah *broadcast domain* .

2.1.1 Broadcast Control

Broadcast terjadi pada setiap *protocol*, bagaimana *broadcast* terjadi dipengaruhi oleh 3 hal berikut:

1. Tipe dari *protocol*
2. Aplikasi yang berjalan pada *internetwork*
3. Bagaimana *service* digunakan

Sejak *switch* banyak memberikan kemudahan, perusahaan-perusahaan mulai mengganti *flat hub networks* dengan *switched network* dan VLAN. Setiap *devices* yang menjadi anggota VLAN mempunyai *broadcast domain* yang sama. Jadi secara *default*, *port* yang tidak menjadi anggota VLAN yang sama, tidak akan menerima *broadcast* tersebut. Hal ini akan mengatasi banyak masalah yang timbul.

2.1.2 Keamanan

Flat internetwork mempunyai kelemahan sebagai berikut:

1. Setiap orang yang menghubungkan *physical network* dapat mengakses *network resources* pada LAN tersebut.
2. Setiap orang dapat menganalisa paket-paket yang terjadi dengan menghubungkan *network analyzer* pada *hub* yang tersedia.

3. *User* dapat bergabung dengan *workgroup* dengan menghubungkan *workstation* pada *hub*.

Hal tersebut membuat VLAN sangat dibutuhkan. Jika kita membuat banyak *broadcast group*, kita dapat mengontrol setiap *port* dan *user*. Sehingga setiap orang yang menghubungkan *workstation* ke *switch port* tidak akan mendapat akses penuh pada jaringan tersebut.

2.1.3 Flexibilitas

Kita tahu bahwa *layer* 2 *switch* hanya membaca *frame* untuk *filtering*, tidak berdasar *network layer*. Dan secara defult *switch* mem-*forward* semua *broadcast*. Tetapi dengan adanya VLAN, kita dapat membuat *broadcast* yang lebih kecil pada *layer* 2.

Setiap *broadcast* yang dikirim, tidak akan dikirim ke VLAN yang berbeda. Dengan menetapkan *switch port* menjadi VLAN *group*, kita mendapat flexibilitas. Flexibilitas dalam menambah *user* pada *broadcast*, tidak peduli lokasi fisik *user* tersebut.

Keunggulan lainya adalah jika anggota VLAN terlalu banyak, kita dapat membuat VLAN untuk menjaga *broadcast* tidak memakan banyak *bandwith*. Semakin sedikit *user* maka semakin sedikit *broadcast* yang diterima sehingga *bandwith* semakin kecil. (Lammle, 2007)

2.2 Web Server

Web server adalah software yang menjadi tulang belakang dari *world wide web* yang berfungsi untuk melayani permintaan halaman-halaman *web*, seperti

website. *Web* server menunggu permintaan dari *client* yang menggunakan *browser* seperti *Netscape Navigator*, *Internet Explorer*, *Modzilla*, dan program *browser* lainnya. Jika ada permintaan dari *browser*, maka *web server* akan memproses permintaan itu kemudian memberikan hasil prosesnya berupa data yang diinginkan kembali ke *browser*. Data ini mempunyai format yang standar, disebut dengan format SGML (*standar general markup language*). Data yang berupa format ini kemudian akan ditampilkan oleh *browser* sesuai dengan kemampuan *browser* tersebut. Contohnya, bila data yang dikirim berupa gambar, *browser* yang hanya mampu menampilkan teks (misalnya *lynx*) tidak akan mampu menampilkan gambar tersebut, dan jika ada akan menampilkan alternatifnya saja.

Web server, untuk berkomunikasi dengan *client*-nya (*web browser*) mempunyai *protocol* sendiri, yaitu HTTP (*hypertext transfer protocol*). Dengan *protocol* ini, komunikasi antar *web server* dengan *client*-nya dapat saling dimengerti dan lebih mudah. Seperti telah dijelaskan diatas, format data pada *world wide web* adalah SGML. Tapi para pengguna *internet* saat ini lebih banyak menggunakan format HTML (*hypertext markup language*) karena penggunaannya lebih sederhana dan mudah dipelajari. Kata *HyperText* mempunyai arti bahwa seorang pengguna *internet* dengan *web browser*nya dapat membuka dan membaca dokumen-dokumen yang ada dalam komputernya atau bahkan jauh tempatnya sekalipun. Hal ini memberikan cita rasa dari suatu proses yang tridimensional, artinya pengguna *internet* dapat membaca dari satu dokumen ke dokumen yang lain hanya dengan meng-klik beberapa bagian dari halaman-halaman dokumen (*web*) itu. Proses yang dimulai dari permintaan *webclient* (*browser*), diterima *web* server, diproses, dan dikembalikan hasil prosesnya oleh *web* server ke *web client*

lagi dilakukan secara transparan. Setiap orang dapat dengan mudah mengetahui apa yang terjadi pada tiap-tiap proses. Secara garis besarnya *web server* hanya memproses semua masukan yang diperolehnya dari *web client*nya. (Budhi,2010)

2.3 HTML (*Hypertext Markup Language*)

HTML merupakan protokol yang digunakan untuk mentransfer data antara *web* server ke *web* browser. *Protocol* ini mentransfer dokumen-dokumen *web* yang ditulis atau berformat HTML. HTML dikatakan *markup language* kaena HTML berfungsi untuk memformat file dokumen teks biasa untuk ditampilkan pada *web* browser dengan bantuan tanda-tanda yang sudah ditentukan. Hal tersebut dapat dilakukan dengan menambahkan elemen atau yang sering disebut sebagai *tag*.

Elemen HTML biasanya berupa tag yang berpasangan dan setiap tag ditandai dengan symbol < dan >. Pasangan dari sebuah tag ditandai dengan tanda '/'. Misalnya pasangan dari tag <contoh> adalah </contoh>. Dalam hal ini <contoh> disebut sebagai elemen dan biasanya dalam suatu elemen terdapat atribut-atribut untuk mengatur elemen tersebut. Contoh elemen yang disertai atribut adalah sebagai berikut: <contoh atribut1='nilai atribut1' atribut2='nilai atribut2'>. Dalam penulisan tag HTML tidaklah case sensitive yang artinya penulisan huruf besar dan kecil tidaklah menjadi masalah.

2.3.1 Struktur dasar HTML

Script HTML dituliskan dalam *text editor* seperti Notepad, kemudian ditulis dengan ekstensi .htm atau .html. kemudian untuk mencobanya dapat dapat

langsung dibuka dengan *web* browser. Setiap dokumen HTML memiliki struktur dasar atau susunan file sebagai berikut:

```
<html>
<head>
<title>berisi teks yang akan muncul pada title bar
browser</title>
</head>
<body>
Berisi tentang teks, gambar atau apapun yang ingin ditampilkan
</body>
</html>
```

Seperti dapat dilihat struktur file HTML diawali dengan sebuah tag <HTML> dan ditutup dengan tag </HTML>. Didalam tag ini terdapat dua bagian besar yaitu yang diapit oleh tag <head>….</head> dan yang diapit oleh tag <body>…</body>.

Bagian yang diapit oleh tag HEAD merupakan *header* dari halaman HTML dan tidak ditampilkan pada window browser. Bagian ini berisi tag-tag *header* seperti <title>…</title> yang berfungsi untuk mengeluarkan judul pada title bar window *web* browser.

Bagian kedua yang diapit oleh tag BODY merupakan bagian yang akan ditampilkan pada window *web* browser nantinya. Pada bagian ini dapat dituliskan semua jenis informasi yang berupa teks dengan bermacam format maupun gambar yang nantinya akan disampaikan pada pembaca. (Sunarfrihantono,2002)

2.4. PHP

PHP adalah bahasa *scripting server-side*, artinya bahasa yang digunakan pada server dengan tanpa perlu melakukan kompilasi tetapi cukup menuliskan tulisan dalam bentuk ASCII-nya saja. PHP sangat mirip dengan penulisan bahasa C, juga mempunyai karakteristik yang mirip dengan perl. Bahasa PHP ini sangat mudah untuk dipelajari.

PHP memungkinkan kita membuat halaman *web* secara *on-the-fly* artinya program pada halaman *web* dapat dibaca oleh orang yang browsing ke server kita. Kita dapat melakukan hal ini dengan cara mengambil data dari database atau file, memanipulasi data tersebut, dan mengirimkan data tersebut ke *web browser*.

Dengan menggunakan PHP, kita dapat meng-*update database*, membuat *database*, dan melakukan perhitungan matematis (termasuk fungsi *trigonometri* yang kompleks). Kita juga dapat membuat dan menghapus file-file secara acak di system kita, bergantung pada level sekuriti yang menjalankan PHP. Kita juga dapat membuat koneksi jaringan *internet* dan melayani koneksi tersebut. Secara teori, kita juga dapat membuat sebuah *web server* yang canggih menggunakan PHP. (Purwanto, 2002)

PHP dapat juga digunakan untuk mengakses sistem. Perintah PHP yang digunakan adalah perintah SHELL_EXEC(*perintah*) dimana perintah yang dimasukan adalah perintah sistem.

2.5 *Shell*

Shell pada *Linux* dapat dianalogikan seperti prompt pada sistim operasi DOS. *Shell* pada *Linux* digunakan untuk melakukan interaksi dengan sistem operasi itu sendiri, dengan cara mengetikan sebuah baris perintah (command line), baik berupa peritah tunggal atau perintah majemuk dan diakhiri tombol enter. Namun cara ber*inter*aksi seperti ini tidaklah efektif, terutama jika dibutuhkan berbagai operasi *shell* untuk melakukan suatu operasi tertentu secara peridoik dimana harus mengetikkan puluhan atau ratusan perintah secara berulang-ulang.

Pada *Linux*, untuk menanggulangi kondisi seperti diatas, dapat dibuat sebuah *script*, yaitu suatu file yang berisi urutan-urutan perintah untuk melakukan suatu operasi tertentu secara sekuensial. Pembuatan sebuah *script* identik dengan pembuatan sebuah program. (Syahputra,2002)

2.6 Ubuntu

Linux adalah salah satu sistem operasi yang menyita banyak perhatian para pengguna komputer yang ditemukan oleh Linus Torvalds. Tak heran jika *Linux* dipakai menjadi nama operating system seperti nama penemunya. Agustus 1991, Linus mengerjakan versi 0.01. Dan pada tahun yang sama, tepatnya pada tanggal 5 October 1991, secara resmi meluncurkan versi 0.02.

Sejarah *Linux* Ubuntu. Ubuntu berasal dari bahasa Afrika yakni "*Humanity to Others*" yang berarti "Kemanusiaan Untuk Sesama". Atas dasar itulah diluncurkannya *Linux* Ubuntu yang dirilis pada tahun 2004. Sistem operasi ini adalah merupakan turunan dari sistem operasi *Linux* yang lain, yakni Debian.

Ubuntu itu sendiri dibuat dengan tujuan bahwa, Ubuntu selalu gratis tanpa adanya biaya lisensi, bersifat *open source* (kode terbuka), dan siap untuk dipergunakan dalam kondisi yang stabil. Ubuntu didukung oleh perusahaan bernama Canonical, Ltd yang memiliki tujuan untuk membantu perkembangan, distribusi, dan promosi dari produk-produk yang bersifat *open source* (kode terbuka). Perusahaan ini bermarkas di Eropa dan dipimpin oleh seseorang bernama Mark Shuttleworth.

Sejak pertama kali diluncurkan, Ubuntu mendapat perhatian yang sangat besar dari pengguna *Linux* yang lain. Hal ini disebabkan karena kestabilan yang

dimiliki oleh Ubuntu itu sendiri. Selain itu kenyamanan dan kemudahan yang dimiliki Ubuntu menjadi daya tarik yang besar bagi pengguna *Linux* di seluruh belahan dunia. (Bunga,2008)

2.7 *Router*

Router berfungsi sebagai penghubung antar dua atau lebih jaringan untuk meneruskan data dari satu jaringan ke jaringan lainnya. *Router* berbeda dengan *switch*. *Switch* merupakan penghubung beberapa alat untuk membentuk suatu *Local Area Network* (LAN). Sebagai ilustrasi perbedaan fungsi dari *Router* dan *switch* merupakan suatu jalanan, dan *Router* merupakan penghubung antar jalan. Masing-masing rumah berada pada jalan yang memiliki alamat dalam suatu urutan tertentu. Dengan cara yang sama, *switch* menghubungkan berbagai macam alat, dimana masing-masing alat memiliki alamat IP sendiri pada sebuah LAN.

Router sangat banyak digunakan dalam jaringan berbasis teknologi protokol TCP/IP, dan *Router* jenis itu disebut juga dengan IP *Router*. Selain IP *Router*, ada lagi *AppleTalk Router*, dan masih ada beberapa jenis *Router* lainnya. Internet merupakan contoh utama dari sebuah jaringan yang memiliki banyak *Router* IP. *Router* dapat digunakan untuk menghubungkan banyak jaringan kecil ke sebuah jaringan yang lebih besar, yang disebut dengan inter*network*, atau untuk membagi sebuah jaringan besar ke dalam beberapa sub*network* untuk meningkatkan kinerja dan juga mempermudah manajemennya. *Router* juga kadang digunakan untuk mengoneksikan dua buah jaringan yang menggunakan media yang berbeda (seperti halnya *Router wireless* yang pada umumnya selain ia dapat menghubungkan komputer dengan menggunakan radio, ia juga mendukung

penghubungan komputer dengan kabel UTP), atau berbeda arsitektur jaringan, seperti halnya dari *Ethernet* ke *Token Ring*.

Gambar 2.3 *Router*
http:// www.router-switch.com/_Cisco/2010/07/02/*Router*/

Router juga dapat digunakan untuk menghubungkan LAN ke sebuah layanan telekomunikasi seperti halnya telekomunikasi leased line atau *Digital Subscriber Line* (DSL). *Router* yang digunakan untuk menghubungkan LAN ke sebuah koneksi leased line seperti T1, atau T3, sering disebut sebagai access server. Sementara itu, *Router* yang digunakan untuk menghubungkan jaringan lokal ke sebuah koneksi DSL disebut juga dengan DSL *Router*. *Router-Router* jenis tersebut umumnya memiliki fungsi firewall untuk melakukan penapisan paket berdasarkan alamat sumber dan alamat tujuan paket tersebut, meski beberapa *Router* tidak memilikinya. *Router* yang memiliki fitur penapisan paket disebut juga dengan *packet-filtering Router*. *Router* umumnya memblokir lalu lintas data yang dipancarkan secara *broadcast* sehingga dapat mencegah adanya *broadcast storm* yang mampu memperlambat kinerja jaringan. (Utomo,2011)

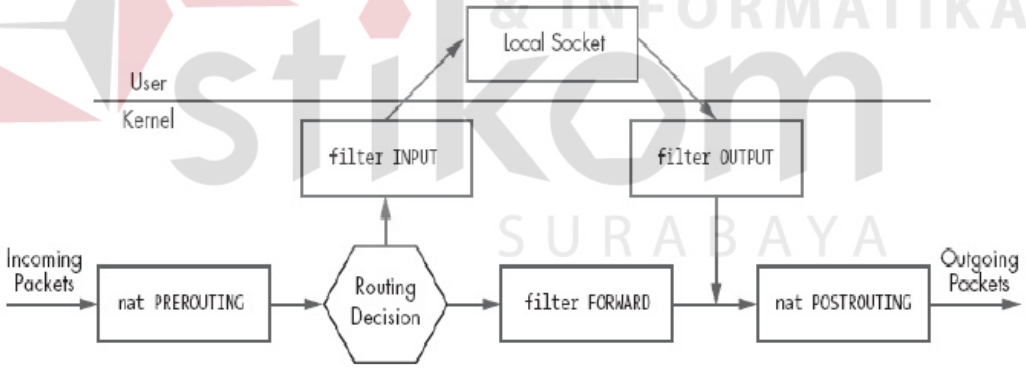
2.8 IP Tables

IPTABLES adalah modul di *Linux* yang memberikan dukungan langsung untuk keamanan sistem serta beberapa keperluan jaringan lainnya. *IPTABLES*

dapat digunakan untuk melakukan seleksi terhadap paket-paket yang datang baik *input*, *output*, maupun *forward* berdasarkan IP *address*, identitas jaringan, nomor *port*, *source* (asal), *destination* (tujuan), *protocol* yang digunakan, bahkan berdasarkan tipe koneksi terhadap setiap paket (data) yang diinginkan.

IP*Tables* mengizinkan *user* untuk mengontrol sepenuhnya jaringan melalui paket IP dengan system *LINUX* yang diimplementasikan pada kernel *Linux*. Sebuah kebijakan atau *Policy* dapat dibuat dengan IPTables.

Sebuah *policy* pada *IPTables* dibuat berdasarkan sekumpulan peraturan yang diberikan pada kernel untuk mengatur setiap paket yang datang. Pada *IPTable* ada istilah yang disebut dengan *IPChain* yang merupakan daftar aturan bawaan dalam *IPTables*. Ketiga *chain* tersebut adalah *INPUT*, *OUTPUT* dan *FORWARD*. Berikut ini adalah diagram perjalanan paket data pada *IPTables*.



Gambar 2.4 blok diagram perintah IP Tables
Sumber : http://www.scribd.com/doc/25831631/IPtables

Pada diagram tersebut, persegi panjang yaitu filter *INPUT*, *filter OUTPUT*, dan *filter FORWARD* menggambarkan ketiga rantai atau *chain*. Pada saat sebuah paket sampai pada salah satu persegipanjang diantara *IPChains*, maka disitulah terjadi proses penyaringan. Rantai akan memutuskan nasib paket tersebut. Apabila keputusannnya adalah *DROP*, maka paket tersebut akan di-*drop*. Tetapi jika rantai

memutuskan untuk *ACCEPT*, maka paket akan dilewatkan melalui diagram tersebut.

Sebuah rantai adalah aturan-aturan yang telah ditentukan. Setiap aturan menyatakan "jika paket memiliki informasi awal (*header*) seperti ini, maka inilah yang harus dilakukan terhadap paket". Jika aturan tersebut tidak sesuai dengan paket, maka aturan berikutnya akan memproses paket tersebut. Apabila sampai aturan terakhir yang ada, paket tersebut belum memenuhi salah satu aturan, maka kernel akan melihat kebijakan bawaan (*default*) untuk memutuskan apa yang harus dilakukan kepada paket tersebut. Ada dua kebijakan bawaan yaitu *default DROP* dan *default ACCEPT*.

Berikut ini adalah format penulisan iptables pada Ubuntu server.

```
iptables [- t tables] [option] [rule] [target]
```

Option terdiri dari perintah, dan parameter serta opsi tambahan, berikut ini adalah daftar dari perintah dan keteranganya:

a. -A atau – *append* Melakukan penambahan *rule*

b. -D atau – *delete* Melakukan penghapusan *rule*

c. -R atau – *replace* Melakukan penggantian *rule*

d. -L atau – *list* Menampilkan ke *display*, daftar iptables

e. -F atau – *flush* Menghapus daftar iptables/pengosongan

f. -I atau – *insert* Melakukan penyisipan *rule*

g. -N atau – *new-chain* Melakukan penambahan *chain* baru

h. -X atau – *delete-chain* Melakukan penghapusan *chain*

i. -P atau – *policy* Memberikan *rule* standard

j. -E atau – rename Memberikan penggantian nama

k. -h atau – help Menampilkan fasilitas menampilkan bantuan

Berikut ini adalah daftar perintah parameter. Parameter digunakan untuk tujuan spesifikasi dari aturan pada iptables tersebut:

a. -p,– *protocol* (*proto*) Parameter ini untuk menentukan perlakuan terhadap *protocol*

b. -s,– *source* (address) Parameter untuk menentukan asal paket

c. -d, – *destination* (address) Parameter untuk menentukan tujuan paket

d. -i, – in-interface paket Masuk melalui interface

e. -o,– out-interface paket keluar melalui interface

f. -s*port-source-port* Menentukan *port* asal

g. -d*port-destination-port* Menentukan *port* tujuan

Berikut ini adalah daftar perintah dari *chain*, *chain* digambarkan sebagai jalur aliran data. *Chain*s yang dipelrukan untuk iptables antara lain:

a. *FORWARD*, yaitu route paket akan di *FORWARD* tanpa diproses lanjut di lokal sistem.

b. *INPUT* yaitu route paket masuk ke dalam lokal sistem.

c. *OUTPUT* yaitu route paket keluar dari lokal sistem.

d. *PREROUTING* yaitu *Chain* yang digunakan untuk keperluan sebelum paket masuk route. Biasanya dipakai untuk proses NAT.

e. *POSTROUTING* yaitu *chain* yang digunakan untuk keperluan perlakuan paket sesudah paket masuk route. Biasanya dipakai untuk proses NAT

Target adalah tujuan perlakuan terhadap *rule* sebelumnya. Pada target terletak keputusan paket data mau ditolak,diteruskan atau diolah terlebih dahulu. Berikut adalah daftar table target iptables.

a. *ACCEPT* Rantai paket tersebut diterima dalam *rule*

b. *DROP* Rantai paket tersebut dibuang

c. *REJECT* Rantai paket tersebut ditolak seperti *DROP*

d. *DNAT* Rantai paket di "*destination* nat" kan ke address lain

e. *SNAT* Rantai paket di arahkan ke *source* nat tertentu

f. *REDIRECT* Rantai paket di redirect ke suatu addres dan *port* tertentu

g. *MASQUERADE* Bekerja seperti SNAT tapi tidak memerlukan *source*

(Scribd, 2011)

2.9 Crontab

Penjadwalan merupakan salah satu tugas seorang sistem administrator, yang bertujuan untuk mengotomatisasi pekerjaan agar dapat berjalan tanpa penekanan tombol keyboard. Dengan adanya program cron pada linux memungkinkan untuk melakukan penjadwalan program yang secara periodik dapat dijalankan, contohnya kita ingin melakukan backup file setiap jam 2 dini hari. Layanan cron merupakan proses yang dibentuk oleh daemon cron. Semua yang dijalankan, diketahui oleh cron dengan membaca entri yang ada pada crontab.

Berikut ini adalah format penulisan Crontab pada Ubuntu server.

```
iptables [menit] [jam] [tanggal] [bulan] [hari] [perintah]
```

a. menit bernilai antara 0 sampai dengan 59.

b. jam bernilai antara 0 sampai dengan 23.

c. tanggal bernilai antara 1 sampai dengan 31.

d. bulan bernilai antara 1 sampai dengan 12.

e. hari bernilai antara 1 sampai dengan 7, nilai 1 berarti hari senin, 2 berarti hari selasa dan seterusnya.

Ada beberapa cara untuk menuliskan *rules* waktu penjadwalan:

1. Menggunakan operator koma (,)

   Dalam *rule* penjadwalan, operator koma diartikan sebagai dan. Berikut ini contoh penulisan dengan operator koma

   contoh: * 2,14 * * * (bintang pertama mewakilin menit, kedua mewakili jam, ketiga mewakili, tanggal, keempat mewakili bulan dan kelima mewakili tahun)

   Rule ini akan menghapus/menambah iptables setiap jam 02.00 dan 14.00 setiap harinya

2. Menggunakan operator asterik (*)

   Operator ini digunakan untuk mengeksekusi perintah setiap waktu

   Contoh: * * * * *

   Rule ini akan menghapus/menambah iptables setiap menit setiap hari

3. Menggunakan operator slash (/)

   Operator "/" digunakan untuk mengeksekusi perintah setiap kelipatan waktu yang ditentukan

   Contoh: */3 * * * *

2.9.1 Membuat penjadwalan dengan crontab

Untuk melakuakn penjadwalan kita dapat menggunakan perintah Crontab, dan setiap melakukan penjadwalan dengan crontab berarti proses tersebut akan diletakkan di dalam /var/spool/cron/nama_*user*. Untuk mengedit file crontab, gunakan perintah berikut :$ crontab –e. Gunakan editor vi untuk memasukkan

perintah yang akan dijadwal. Untuk melihat isi crontab dapat dilakuakn dengan perintah : $ crontab –1. (ICT center,2005)

2.10 TCP DUMP

TCP/IP merupakan standar untuk komunikasi antara dua komputer atau lebih. IP (Internet *Protocol*) menjalankan fungsinya pada *network* layer (pengalamatan dan *routing*) sedangkan TCP (*Transmission Control Protocol*) menyediakan jalur hubungan *end-to-end* (*transport* layer).

TCPdump adalah tools yang berfungsi menangkap, membaca paket yang sedang ditransmisikan melalui jalur TCP. Untuk menjalankan TCPdump anda harus menggunakan *user* root. Adapun sintaks untuk menjalankan TCPdump adalah sebagai berikut

```
TCPdump [ -adeflnNOpqRStuvxX ] [ -c count ] [ -C file_size ] [ -F
file ][ -i interface ] [ -m module ] [ -r file ] [ -s snaplen ] [
-T type ] [ -w file ] [ -E algo:secret ] [ expression ]
```

Contoh : # TCPdump –i eth0

TCPdump –X –i eth0

TCP dump –n –v –i eth0

Perintah tersebut digunakan untuk melihat semua *header* paket yang termonitor oleh interface eth0. Pada perintah yang ke dua, selain *header* paket akan di tampilkan isi data yang di bawa dalam paket yang termonitor oleh interface eth0 baik dalam bentuk *binary (hexadesimal)* maupun dalam bentuk ASCII. Perintah ketiga berguna untuk melihat paket yang keluar masuk, dengan menghilangkan paket tertentu (-n) dan menambah kelengkapan informasi paket (-v)

Berikut adalah cuplikan hasil dari TCP dump:

```
[cc lang="VHDL"]19:57:06.748557 IP (tos 0×0, ttl 64, id 33646,
offset 0, flags [DF], proto TCP (6), length 60)
192.168.1.4.33922 > 68.178.254.190.80: Flags [S], cksum 0x83ae
(correct), seq 4011514848, win 5840, options [mss 1460,sackOK,TS
val 612494 ecr 0,nop,wscale 6], length 0[/cc]
```

Keterangan:

1. 19:57:06.748557 merupakan waktu diambilnya TCPdump
2. tos 0×0 => tipe servis
3. ttl 64 merupakan *time* to live
4. id 33646 menunjukan nomer id paket
5. [DF] berarti don't *fragment*, artinya paket yang dikirim tidak terbagi-bagi.
6. *proto* TCP merupakan tipe dari *protocol*, bisa UDP, TCP ataupun ICMP
7. *length* 60 panjang paket + *header*
8. 192.168.1.4.33922 192.168.1.4 merupakan ip sumber dan 33922 *port number* yang digunakan oleh *client*
9. 68.178.254.190.80 , ip tujuan dengan *port* 80
10. *Flag*s [S] merupakan *flag* dari TCP dimana parameter S berarti ack *reply* dari server, parameter r berarti koneksi diulang, parameter F berarti untuk penanda akir dari pengiriman data, dan parameter P berarti data *harus* dikirim secepatnya.
11. cksum 0x83ae (correct) merupakan TCP-*header* check-sum dari paket.
12. seq 4011514848 merupakan TCP sequence *number*
13. win 5840 merupakan jumlah yang akan dikirim sebelum mendapat Paket ACK dikirim kembali dari server
14. *length* 0 merupakan panjang dari data, bernilai 0 karena yang dikirim adalah *header*. (Onno, 2002)