BAB III

LANDASAN TEORI

3.1 Firewall

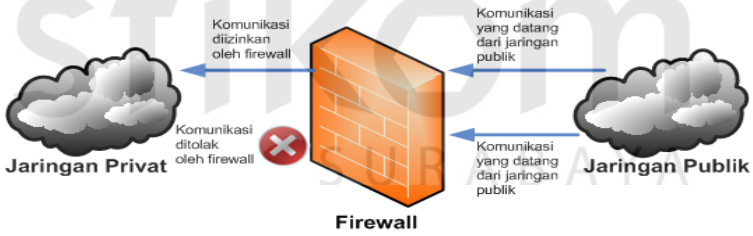
Firewall merupakan salah satu cara yang efektif unutk melindungi system dari ancaman terhadap keamanan jaringan komputer. Firewall perlu diterapkan karena dapat meningkatkan *host security* dari serangan para cracker bahkan *hacker* melalui celah-celah yang terbuka, seperti protocol UDP/TCP (panduan lengkap jaringan komputer, Lukmana, Lukas., 2004). Umumnya, sebuah firewall diimplementasikan dalam sebuah mesin terdedikasi, yang berjalan pada pintu gerbang (*gateway*) antara jaringan lokal dan jaringan lainnya. Firewall umumnya juga digunakan untuk mengontrol akses terhadap siapa saja yang memiliki akses terhadap jaringan pribadi dari pihak luar. Saat ini, istilah firewall menjadi istilah generik yang merujuk pada sistem yang mengatur komunikasi antar dua jaringan yang berbeda. Mengingat saat ini banyak perusahaan yang memiliki akses ke internet dan juga tentu saja jaringan korporat di dalamnya, maka perlindungan terhadap aset digital perusahaan tersebut dari serangan para *hacker*, pelaku *spionase*, ataupun pencuri data lainnya, menjadi esensial.

Dalam sebuah paket data terdiri dari dua bagian, yakni bagian *header* yang berisi informasi yang berhubungan dengan paket tersebut seperti asal paket, tujuan paket, tipe paket, dan beberapa informasi lainnya tentang paket yang dibawa. Bagian lainnya yaitu bagian *body* yang berisi data yang akan dikirim dalam paket tersebut. Firewall berisi sederetan daftar aturan (*rule*) yang digunakan untuk memeriksa asal, tujuan, port, dan informasi tipe data paket yang

terdapat pada masing-masing paket, dimana aturan tersebut akan menentukan nasib dapat atau tidaknya sebuah paket masuk kedalam sebuah jaringan komputer. Firewall dapat bersifat sangat terbuka jika dikonfigurasikan untuk mengizinkan hampir semua layanan internet bebas keluar masuk dari dan keluar jaringan. Tetapi dapat juga sebaliknya, sebuah firewall dapat bersifat tertutup jika dikonfigurasikan dengan aturan (*rule*) yang sangat ketat dan akses yang sangat terbatas.

Beberapa Fungsi dipergunakannya firewall dalam jaringan adalah sebagai berikut:

1. Mengatur dan mengontrol lalu lintas jaringan
2. Melakukan autentikasi terhadap akses
3. Melindungi sumber daya dalam jaringan privat
4. Mencatat semua kejadian, dan melaporkan kepada administrator



Gambar 3.1. Ilustrasi Firewall dalam jaringan

Firewall memiliki banyak jenis yang bisa diaplikasikan pada komputer-komputer kantor maupun server atau komputer pribadi maka agar lebih mudah dapat disederhanakan menurut jumlah komputer yang menggunakan firewall.

Secara umum firewall dapat dibagi atas dua jenis, yakni:

1. **Personal Firewall**

   Personal Firewall didesain untuk melindungi sebuah komputer yang terhubung ke jaringan dari akses yang tidak dikehendaki. Firewall jenis ini akhir-akhir ini berevolusi menjadi sebuah kumpulan program yang bertujuan untuk mengamankan komputer secara total, dengan ditambahkannya beberapa fitur pengaman tambahan semacam perangkat proteksi terhadap virus, anti-spyware, anti-spam, dan lainnya. Bahkan beberapa produk firewall lainnya dilengkapi dengan fungsi pendeteksian gangguan keamanan jaringan (*Intrucsion Detection System*). Contoh dari firewall jenis ini adalah Microsoft Windows Firewall (yang telah terintegrasi dalam sistem operasi Windows XP Service Pack 2, Windows Vista dan Windows Server 2003 Service Pack 1), Symantec Norton Personal Firewall, Kerio Personal Firewall, dan lain-lain. *Personal Firewall* secara umum hanya memiliki dua fitur utama, yakni *packet filter firewall* dan *stateful firewall*.

2. **Network Firewall**

   Network Firewall didesain untuk melindungi jaringan secara keseluruhan dari berbagai serangan. Umumnya dijumpai dalam dua bentuk, yakni sebuah perangkat terdedikasi atau sebagai sebuah perangkat lunak yang diinstalasikan dalam sebuah *server*. Contoh dari firewall ini adalah Microsoft Internet Security and Acceleration Server (ISA Server), Cisco

PIX, Cisco ASA, IPTables dalam sistem operasi GNU/Linux, pf dalam keluarga sistem operasi Unix BSD, serta SunScreen dari Sun Microsystems, Inc. yang dibundel dalam sistem operasi Solaris. *Network Firewall* secara umum memiliki beberapa fitur utama, yakni apa yang dimiliki oleh *Personal Firewall* (paket *filter firewall* dan *stateful firewall*), *circuit level gateway*, *application level gateway*, dan juga *NAT firewall*. *Network firewall* umumnya bersifat transparan (tidak terlihat) dari pengguna dan menggunakan teknologi *routing* untuk menentukan paket mana yang diizinkan, dan mana paket yang akan ditolak.

Pada firewall terjadi beberapa proses yang memungkinkan firewall dapat melindungi jaringan. Proses yang terjadi pada firewall ada tiga macam, yaitu :

1. Modifikasi header paket

   Digunakan untuk memodifikasi kualitas layanan bit paket TCP sebelum terjadi *routing*.

2. Translasi alamat jaringan

   Translasi alamat jaringan yang terjadi antara jaringan *private* dan jaringan *public* dapat berupa translasi satu ke satu (*one to one*) yang mana satu alamat IP *private* dipetakan ke satu alamat IP *public*, serta translasi banyak ke satu (*many to one*) yang mana beberapa alamat IP *private* dipetakan ke satu alamat IP *public*

3. Packet Filtering

   Berbagai kebijakan dapat diterapkan dalam melakukan operasi *packet-filtering*. Pada intinya, berupa mekanisme pengontrolan data yang diperbolehkan mengalir dari dan / atau ke jaringan internal, dengan

menggunakan beberapa parameter yang tercantum dalam *header* paket data: arah (*inbound* atau *outbound*), *address* asal dan tujuan, port asal dan tujuan, serta jenis protokol transport. *Router* akan mengevaluasi informasi ini dalam setiap paket data yang mengalir melaluinya, kemudian menetapkan aksi yang harus dilakukan terhadap paket tersebut, berdasarkan set aturan / program dalam *packet-filtering*. Sehingga keputusan *routing* dasar *router* tersebut, kemudian dilengkapi dengan bagian dari kebijakan sekuriti jaringan.

Tabel berikut akan menunjukan contoh konfigurasi operasi *packet-filtering*, untuk menyediakan hanya fasilitas SMTP *inbound* dan *outbound* pada jaringan.

Tabel.3.1 Contoh konfigurasi operasi paket filtering

Aturan	**Arah**	**Address Asal**	**Address Tujuan**	**Protokol**	**Port Tujuan**	**Aksi**
A	masuk	eksternal	internal	TCP	25	ok
B	keluar	internal	eksternal	TCP	>1023	ok
C	keluar	internal	eksternal	TCP	25	ok
D	masuk	eksternal	internal	TCP	>1023	ok
E	sembarang	sembarang	sembarang	sembarang	sembarang	deny

Aturan A dan B melayani hubungan SMTP *inbound* (email datang), aturan C dan D melayani hubungan SMTP *outbound* (email keluar) serta aturan E merupakan aturan *default* yang dilakukan bila aturan aturan sebelumnya gagal. Kalau diamati lebih dekat, selain trafik SMTP konfigurasi tersebut juga masih membolehkan hubungan masuk dan keluar pada port >1023 (aturan B dan D), sehingga terdapat kemungkinan bagi program-program *server* seperti X11 (port 6000), OpenWindows (port 2000), atau kebanyakan program basis-data (Sybase, *Oracle, Informix*, dll), untuk dihubungi dari luar. Untuk menutup kemungkinan ini, diperlukan evaluasi parameter lain, seperti evaluasi port asal. Dengan cara ini, satu-satunya celah menembus firewall adalah dengan menggunakan port SMTP. Bila kita masih juga kurang yakin dengan kejujuran para pengguna port ini, dapat dilakukan evaluasi lebih lanjut dari informasi ACK.

Salah satu perangkat lunak yang banyak digunakan untuk keperluan proses pada firewall adalah ***Iptables***. Program ini adalah program administratif untuk *packet-filtering* dan NAT (*Network Address Translation*).

Dibawah ini akan dijelaskan berberapa macam cara kerja yang biasanya dipakai untuk firewall:

A. Packet-Filter Firewall

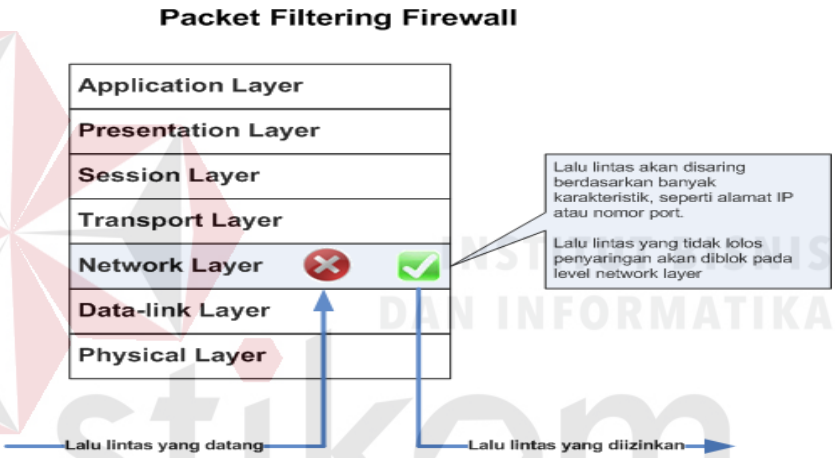
Pada bentuknya yang paling sederhana, sebuah firewall adalah sebuah *router* atau komputer yang dilengkapi dengan dua buah NIC (*Network Interface Card*) yang mampu melakukan penapisan atau penyaringan

terhadap paket-paket yang masuk. Perangkat jenis ini umumnya disebut dengan *packet-filtering router*.

Firewall jenis ini bekerja dengan cara membandingkan alamat sumber dari paket-paket tersebut dengan kebijakan pengontrolan akses yang terdaftar dalam *access control list firewall*, *router* tersebut akan mencoba memutuskan apakah hendak meneruskan paket yang masuk tersebut ke tujuannya atau menghentikannya. Pada bentuk yang lebih sederhana lagi, *firewall* hanya melakukan pengujian terhadap alamat IP atau nama domain yang menjadi sumber paket dan akan menentukan apakah hendak meneruskan atau menolak paket tersebut. Meskipun demikian, *packet-filtering router* tidak dapat digunakan untuk memberikan akses (atau menolaknya) dengan menggunakan basis hak-hak yang dimiliki oleh pengguna.

Packet-filtering router juga dapat dikonfigurasikan agar menghentikan beberapa jenis lalu lintas jaringan dan tentu saja mengizinkannya. Umumnya, hal ini dilakukan dengan mengaktifkan / menonaktifkan port TCP/IP dalam sistem firewall tersebut. Sebagai contoh, port 25 yang digunakan oleh [[|SMTP|Protokol SMTP]] (*Simple Mail Transfer Protocol*) umumnya dibiarkan terbuka oleh beberapa firewall untuk mengizinkan surat elektronik dari internet masuk ke dalam jaringan privat, sementara port lainnya seperti port 23 yang digunakan oleh Protokol Telnet dapat dinonaktifkan untuk mencegah pengguna internet untuk mengakses layanan yang terdapat dalam jaringan privat tersebut.

Firewall juga dapat memberikan semacam pengecualian (*exception*) agar beberapa aplikasi dapat melewati firewall tersebut. Dengan menggunakan pendekatan ini, keamanan akan lebih kuat tapi memiliki kelemahan yang signifikan yakni kerumitan konfigurasi terhadap firewall: daftar *access control list firewall* akan membesar seiring dengan banyaknya alamat IP, nama domain, atau port yang dimasukkan ke dalamnya, selain tentunya juga *exception* yang diberlakukan.



Gambar.3.2. Cara kerja packet filter firewall

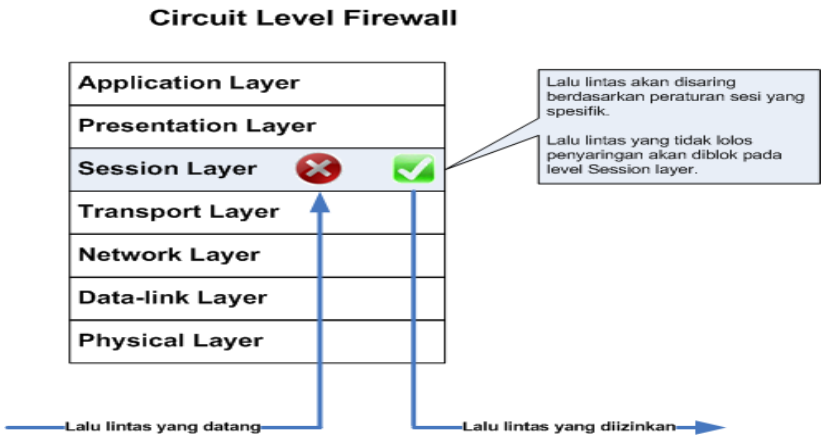
B. Circuit Level Gateway

Firewall jenis lainnya adalah *circuit-level gateway*, yang umumnya berupa komponen dalam sebuah *proxy server*. Firewall jenis ini beroperasi pada level yang lebih tinggi dalam model referensi tujuh lapis OSI (bekerja pada lapisan *session layer*) daripada *packet filter firewall*. Modifikasi ini membuat firewall jenis ini berguna dalam rangka menyembunyikan informasi mengenai jaringan terproteksi, meskipun firewall ini tidak

melakukan penyaringan terhadap paket-paket individual yang mengalir dalam koneksi.

Dengan menggunakan firewall jenis ini, koneksi yang terjadi antara pengguna dan jaringan pun disembunyikan dari pengguna. Pengguna akan dihadapkan secara langsung dengan firewall pada saat proses pembuatan koneksi dan firewall pun akan membentuk koneksi dengan sumber daya jaringan yang hendak diakses oleh pengguna setelah mengubah alamat IP dari paket yang ditransmisikan oleh dua belah pihak. Hal ini mengakibatkan terjadinya sebuah sirkuit virtual (*virtual circuit*) antara pengguna dan sumber daya jaringan yang ia akses.

Firewall ini dianggap lebih aman dibandingkan dengan *packet-filtering firewall,* karena pengguna eksternal tidak dapat melihat alamat IP jaringan internal dalam paket-paket yang ia terima, melainkan alamat IP dari firewall. Protokol yang populer digunakan sebagai *circuit-level gateway* adalah SOCKS v5.



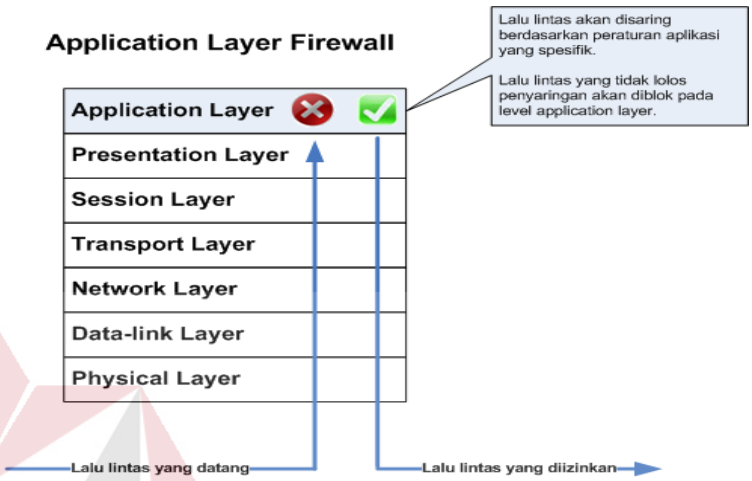
Gambar.3.3. Cara kerja circuit level firewall

C. Application Level Firewall

Firewall jenis lainnya adalah *application level gateway* (atau sering juga disebut sebagai *proxy firewall*), yang umumnya juga merupakan komponen dari sebuah *proxy server*. Firewall ini tidak mengizinkan paket yang datang untuk melewati firewall secara langsung. Tetapi, aplikasi *proxy* yang berjalan dalam komputer yang menjalankan firewall akan meneruskan permintaan tersebut kepada layanan yang tersedia dalam jaringan privat dan kemudian meneruskan respons dari permintaan tersebut kepada komputer yang membuat permintaan pertama kali yang terletak dalam jaringan publik yang tidak aman.

Umumnya, firewall jenis ini akan melakukan autentikasi terlebih dahulu terhadap pengguna sebelum mengizinkan pengguna tersebut untuk mengakses jaringan. Selain itu, firewall ini juga mengimplementasikan mekanisme *auditing* dan pencatatan (*logging*) sebagai bagian dari kebijakan keamanan yang diterapkannya. *application level firewall* juga umumnya mengharuskan beberapa konfigurasi yang diberlakukan pada pengguna untuk mengizinkan mesin klien agar dapat berfungsi. Sebagai contoh, jika sebuah *proxy* FTP dikonfigurasikan di atas sebuah *application layer gateway, proxy* tersebut dapat dikonfigurasikan untuk mengizinlan beberapa perintah FTP, dan menolak beberapa perintah lainnya. Jenis ini paling sering diimplementasikan pada *proxy* SMTP sehingga mereka dapat menerima surat elektronik dari luar (tanpa menampakkan alamat e-mail internal), lalu meneruskan e-mail tersebut kepada *e-mail server* dalam

jaringan. Tetapi, karena adanya pemrosesan yang lebih rumit, firewall jenis ini mengharuskan komputer yang dikonfigurasikan sebagai *application gateway* memiliki spesifikasi yang tinggi, dan tentu saja jauh lebih lambat dibandingkan dengan *packet-filter firewall*.



Gambar.3.4. Application Level Firewall (disebut juga sebagai *application proxy* atau *application level gateway*)

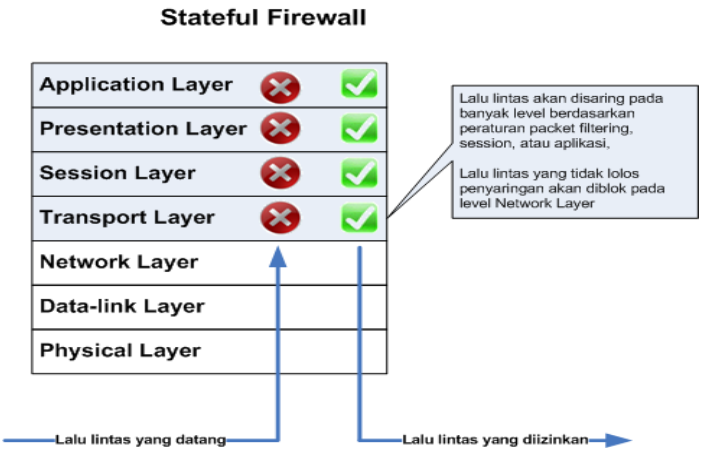
D. NAT Firewall

NAT (*Network Address Translation*) Firewall secara otomatis menyediakan proteksi terhadap sistem yang berada di balik firewall karena NAT Firewall hanya mengizinkan koneksi yang datang dari komputer-komputer yang berada di balik firewall. Tujuan dari NAT adalah untuk melakukan *multiplexing* terhadap lalu lintas dari jaringan internal untuk kemudian menyampaikannya kepada jaringan yang lebih luas (MAN, WAN atau Internet) seolah-olah paket tersebut datang dari sebuah alamat IP atau beberapa alamat IP. NAT Firewall membuat tabel dalam memori yang mengandung informasi mengenai koneksi yang dilihat oleh firewall.

Tabel ini akan memetakan alamat jaringan internal ke alamat eksternal. Kemampuan untuk menaruh keseluruhan jaringan di belakang sebuah alamat IP didasarkan terhadap pemetaan terhadap port-port dalam NAT firewall.

E. Stateful Firewall

Stateful Firewall merupakan sebuah firewall yang menggabungkan keunggulan yang ditawarkan oleh *packet-filtering firewall, NAT firewall, circuit-level firewall* dan *proxy firewall* dalam satu sistem. *Stateful Firewall* dapat melakukan *filtering* terhadap lalu lintas berdasarkan karakteristik paket, seperti halnya *packet-filtering firewall*, dan juga memiliki pengecekan terhadap sesi koneksi untuk meyakinkan bahwa sesi koneksi yang terbentuk tersebut diizinlan. Tidak seperti *proxy firewall* atau *circuit level firewall*, *stateful firewall* umumnya didesain agar lebih transparan (seperti halnya *packet-filtering firewall* atau NAT firewall). Tetapi, *stateful firewall* juga mencakup beberapa aspek yang dimiliki oleh *application level firewall*, sebab ia juga melakukan inspeksi terhadap data yang datang dari lapisan aplikasi (*application layer*) dengan menggunakan layanan tertentu. Firewall ini hanya tersedia pada beberapa firewall kelas atas, semacam Cisco PIX. Karena menggabungkan keunggulan jenis-jenis firewall lainnya sehingga menjadi lebih komplek dari pada firewall-firewall lainnya.



Gambar.3.5. Cara kerja stateful firewall

F. Virtual Firewall

Virtual firewall adalah sebutan untuk beberapa firewall logis yang berada dalam sebuah perangkat fisik (komputer atau perangkat firewall lainnya). Pengaturan ini mengizinkan beberapa jaringan agar dapat diproteksi oleh sebuah firewall yang unik yang menjalankan kebijakan keamanan yang juga unik, cukup dengan menggunakan satu buah perangkat. Dengan menggunakan firewall jenis ini, sebuah ISP (*Internet Service Provider*) dapat menyediakan layanan firewall kepada para pelanggannya, sehingga mengamankan lalu lintas jaringan mereka, hanya dengan menggunakan satu buah perangkat. Hal ini jelas merupakan penghematan biaya yang signifikan, meski firewall jenis ini hanya tersedia pada firewall kelas atas, seperti Cisco PIX 535.

G. Transparent Firewall

Transparent firewall (juga dikenal sebagai *bridging firewall*) bukanlah sebuah firewall yang murni, tetapi ia hanya berupa turunan dari *stateful*

firewall. Daripada firewall-firewall lainnya yang beroperasi pada lapisan IP ke atas, *transparent firewall* bekerja pada lapisan *data-link layer*, dan kemudian ia memantau lapisan-lapisan yang ada di atasnya. Selain itu, *transparent firewall* juga dapat melakukan apa yang dapat dilakukan oleh *packet-filtering firewall*, seperti halnya *stateful firewall* dan tidak terlihat oleh pengguna (karena itulah, ia disebut sebagai ***transparent firewall***).

Intinya, *transparent firewall* bekerja sebagai sebuah *bridge* yang bertugas untuk menyaring lalu lintas jaringan antara dua segmen jaringan. Dengan menggunakan *transparent firewall*, keamanan sebuah segmen jaringan pun dapat diperkuat, tanpa harus mengaplikasikan *NAT filter*.

Transparent firewall menawarkan tiga buah keuntungan, yakni sebagai berikut:

1. Konfigurasi yang mudah (bahkan beberapa produk mengklaim sebagai "*zero configuration*"). Hal ini memang karena *transparent firewall* dihubungkan secara langsung dengan jaringan yang hendak diproteksinya, dengan memodifikasi sedikit atau tanpa memodifikasi konfigurasi firewall tersebut. Karena ia bekerja pada *data-link layer*, pengubahan alamat IP pun tidak dibutuhkan. Firewall juga dapat dikonfigurasikan untuk melakukan segmentasi terhadap sebuah subnet jaringan antara jaringan yang memiliki keamanan yang rendah dan keamanan yang tinggi atau dapat juga untuk melindungi sebuah host, jika memang diperlukan.

2. Kinerja yang tinggi. Hal ini disebabkan oleh firewall yang berjalan dalam lapisan *data-link* lebih sederhana dibandingkan dengan firewall yang berjalan dalam lapisan yang lebih tinggi. Karena bekerja lebih sederhana, maka kebutuhan pemrosesan pun lebih kecil dibandingkan dengan firewall yang berjalan pada lapisan yang tinggi, dan akhirnya performa yang ditunjukannya pun lebih tinggi.
3. Tidak terlihat oleh pengguna (*stealth*). Hal ini memang dikarenakan *transparent firewall* bekerja pada lapisan *data-link*, dan tidak membutuhkan alamat IP yang ditetapkan untuknya (kecuali untuk melakukan manajemen terhadapnya, jika memang jenisnya *managed firewall*). Karena itulah, *transparent firewall* tidak dapat terlihat oleh para penyerang. Karena tidak dapat diraih oleh penyerang (tidak memiliki alamat IP), penyerang pun tidak dapat menyerangnya.

3.2 Server Proxy

Proxy server adalah sebuah komputer *server* atau program komputer yang dapat bertindak sebagai komputer lainnya untuk melakukan *request* terhadap *content* dari *internet* atau *intranet.*

Proxy server bertindak sebagai *gateway* terhadap *internet* atau jaringan eksternal untuk setiap komputer klien. *Proxy server* tidak terlihat oleh komputer klien atau dengan kata lain seorang pengguna yang berinteraksi dengan *internet* melalui sebuah *proxy server* tidak akan mengetahui bahwa sebuah *proxy server* sedang menangani *request* yang dilakukannya. *Web server* yang menerima *request* dari *proxy server* akan menginterpretasikan *request-request* tersebut

seolah-olah *request* itu datang secara langsung dari komputer klien, bukan dari *proxy server*.

Proxy server juga dapat digunakan untuk mengamankan jaringan pribadi yang dihubungkan ke sebuah jaringan publik (seperti halnya *internet*). *Proxy server* memiliki lebih banyak fungsi daripada *router* yang memiliki fitur *packet filtering* karena memang *proxy server* beroperasi pada level yang lebih tinggi dan memiliki kontrol yang lebih menyeluruh terhadap akses jaringan. *Proxy server* yang berfungsi sebagai sebuah "agen keamanan" untuk sebuah jaringan pribadi, umumnya dikenal sebagai *firewall*.

Akses yang dilakukan para pengguna *internet* saat ini kebanyakan melalui protokol HTTP. Walaupun protokol yang lain cukup banyak, tetapi koneksi HTTP masih mendominasi penggunaan jaringan *internet*. Proses transfer file yang dulunya banyak dilakukan menggunakan protokol FTP, sekarang sudah dapat dilakukan menggunakan protokol HTTP.

Kebanyakan para pengguna *internet* mengunjungi situs yang sama secara berulang-ulang. Misalnya suatu situs sudah pernah dikunjungi oleh pengguna tertentu yang lebih dahulu menggunakan jaringan *internet*, mungkin pada kesempatan berikutnya ada pengguna lain yang mengunjungi situs yang sama. Pengguna ini akan berhubungan langsung dengan *server web* yang sama dan ini memerlukan bandwidth sehingga menyebabkan koneksi internet menjadi lambat.

Untuk meningkatkan efisiensi penggunaan bandwidth, biasanya dalam suatu jaringan lokal dipasang suatu *server proxy*. *Server proxy* punya kemampuan untuk melakukan filter situs yang dikunjungi maupun filter terhadap pengunjung,

sehingga *server proxy* dalam hal ini dapat berlaku sebagai suatu *firewall*. Selain itu, *server proxy* juga punya kemampuan untuk menyimpan file-file yang berasal dari situs yang pernah dikunjungi, atau yang disebut dengan *server cache*.

Server proxy adalah server yang diletakkan antara suatu aplikasi klien dan aplikasi server yang dihubungi. Aplikasi klien dapat berupa *web browser*, klien FTP dan sebagainya. Sedangkan aplikasi server dapat berupa *server web*, server FTP. *Server proxy* yang diletakkan di antara aplikasi klien dan server tersebut, dapat digunakan untuk mengendalikan maupun memonitor lalu lintas paket data yang melewatinya.

Secara umum manfaat *server proxy* ada 2 macam, sebagai berikut :

1. Meningkatkan kinerja jaringan.

   Dengan kemampuan *server proxy* untuk menyimpan data permintaan dari *web browser*, permintaan yang sama dengan permintaan sebelumnya hanya akan diambilkan dari simpanan *server proxy*. Jika seorang user sudah pernah membuka suatu situs, maka pengguna berikutnya yang membuka situs yang sama tidak perlu dihubungkan langsung pada situs sumbernya, tetapi cukup diambilkan dari simpanan *server proxy*. Dengan cara demikian, koneksi langsung pada server sumbernya dapat dikurangi sehingga penggunaan bandwidth internet untuk koneksi langsung menjadi berkurang.

2. Filter Permintaan

   Server proxy juga dapat digunakan sebagai *filter* terhadap situs yang boleh atau tak boleh dikunjungi. Selain itu, *server proxy* juga dapat sebagai *filter*

terhadap aplikasi klien yang dapat menggunakan akses terhadap internet. Dalam hal ini *server proxy* berlaku sebagai *filter* terhadap *user internet*.

Karena manfaatnya cukup besar, kebanyakan jaringan privat selalu menggunakan *server proxy* untuk koneksi ke internet. Salah satu perangkat lunak yang sering digunakan sebagai *server proxy* adalah **squid**. Program squid punya kemampuan untuk berlaku sebagai *server cache* atau sebagai *filter*.

Dalam menjalankan fungsinya sebagai *server proxy*, squid biasanya dikombinasikan dengan firewall (*iptables*). Dalam hal ini iptables digunakan untuk mengarahkan komputer klien agar menggunakan *server proxy* dalam berhubungan dengan internet.

Dalam kaitannya dengan firewall, *server proxy* dapat dipasang pada komputer yang sama dengan firewall atau pada komputer yang terpisah dengan firewall. Masing-masing konfigurasi akan menentukan aturan rantai yang diberlakukan pada firewall maupun konfigurasi *server proxy* sendiri.

3.3 Squid

Squid adalah *high-performance proxy caching server* untuk web klien, yang sudah mendukung *FTP*, *ghoper*, dan *HTTP data object*. Berbeda dengan software caching yang lama, squid menangani semua permintaan tunggal (*single*), *non-blocking, I/O-driven proses*. Squid menyimpan meta data dan terutama *hot-object* yang di simpan di RAM, menyimpan *DNS lookups*, mendukung *non-blocking DNS lookups*, dan implementasi *negative-caching* jika permintaan gagal. Squid mendukung SSL, akses kontrol yang banyak, dan *full request logging*. Dengan menggunakan *lightweight internet cache protocol*, *squid cache* dapat

dibuat dalam suatu hirarki atau *mesh* untuk meningkatkan penghematan bandwidth.

Squid terdiri dari program server utama squid, sebuah *Domain Name System lookup* (program DNS server), beberapa program tambahan untuk permintaan menulis ulang dan melakukan *authentication*, dan beberapa *tools management client*. Ketika squid dijalankan, itu akan menambah jumlah proses dnsserver, masing-masing bertugas sendiri-sendiri, *blocking Domain Name System* (DNS) *lookup*. Ini akan mengurangi waktu tunggu *DNS lookups*.

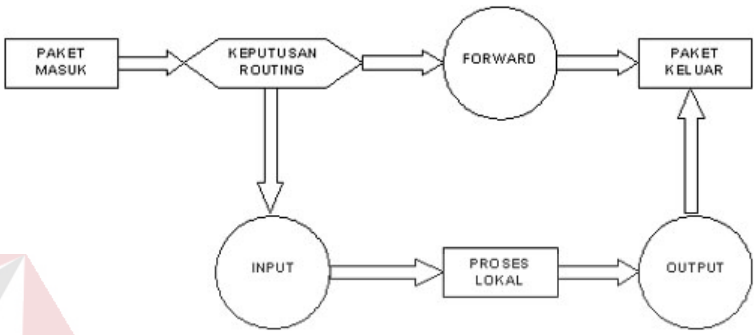
Paket squid terutama berisi daemon untuk *server proxy* yang namanya sama dengan nama paketnya yaitu squid. Selain itu juga sudah disediakan beberapa file konfigurasi *default* yang memungkinkan daemon squid dapat langsung dijalankan.

Langkah-langkah yang diperlukan untuk mengkonfigurasi squid secara umum adalah sebagai berikut:

1. Instal aplikasi squid
2. Edit file konfigurasi squid
3. Buat direktori dan file blacklist untuk *blacklist* secara manual
4. Buat direktori dan file untuk pesan error dalam bahasa indonesia
5. Check konfigurasi yang diterapkan di proxy
6. *Running service*
7. Test konfigurasi di proxy dan *client*

3.4 IP Tables

IPTables memiliki tiga macam daftar aturan bawaan dalam tabel penyaringan, daftar tersebut dinamakan rantai firewall *(firewall chain)* atau sering disebut *chain* saja. Ketiga *chain* tersebut adalah INPUT, OUTPUT dan FORWARD.



Gambar.3.6. Firewall Chain

Pada gambar tersebut, lingkaran menggambarkan ketiga rantai atau *chain*. Pada saat sebuah paket sampai pada sebuah lingkaran, 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.

Jalannya sebuah paket melalui gambar tersebut bisa dicontohkan sebagai berikut:

A. Perjalanan paket yang *diforward* ke host yang lain :

1. Paket berada pada jaringan fisik, contoh internet.
2. Paket masuk ke interface jaringan, contoh eth0.
3. Paket masuk ke *chain PREROUTING* pada table *Mangle*. *Chain* ini berfungsi untuk me-mangle (menghaluskan) paket, seperti merubah TOS, TTL dan lain-lain.
4. Paket masuk ke *chain PREROUTING* pada tabel nat. Chain ini berfungsi utamanya untuk melakukan DNAT *(Destination Network Address Translation).*
5. Paket mengalami keputusan routing, apakah akan diproses oleh host lokal atau diteruskan ke host lain.
6. Paket masuk ke *chain FORWARD* pada tabel filter. Disinlah proses pemfilteran yang utama terjadi.
7. Paket masuk ke *chain POSTROUTING* pada tabel nat. *Chain* ini berfungsi utamanya untuk melakukan SNAT *(Source Network Address Translation).*
8. Paket keluar menuju interface jaringan, contoh eth1.
9. Paket kembali berada pada jaringan fisik, contoh LAN.

B. Perjalanan paket yang ditujukan bagi host lokal :

1. Paket berada dalam jaringan fisik, contoh internet.
2. Paket masuk ke interface jaringan, contoh eth0.
3. Paket masuk ke *chain PREROUTING* pada tabel mangle.
4. Paket masuk ke *chain PREROUTING* pada tabel nat.
5. Paket mengalami keputusan routing.
6. Paket masuk ke *chain INPUT* pada tabel filter untuk mengalami proses penyaringan.
7. Paket akan diterima oleh aplikasi lokal.

C. Perjalanan paket yang berasal dari host lokal

1. Aplikasi lokal menghasilkan paket data yang akan dikirimkan melalui jaringan.
2. Paket memasuki *chain OUTPUT* pada tabel mangle.
3. Paket memasuki *chain OUTPUT* pada tabel nat.
4. Paket memasuki *chain OUTPUT* pada tabel filter.
5. Paket mengalami keputusan routing, seperti ke mana paket harus pergi dan melalui interface mana.
6. Paket masuk ke *chain POSTROUTING* pada tabel NAT.
7. Paket masuk ke interface jaringan, contoh eth0.
8. Paket berada pada jaringan fisik, contoh internet.

Terdapat beberapa cara untuk mengatur ke seluruh *cain*, diantaranya adalah :

1. membuat sebuah *chain* baru (-N)
2. menghapus sebuah *chain* kosong (-X)
3. mengubah kebijakan (*policy*) untuk *chain* built-in (-P)
4. melihat daftar aturan (*rules*) dalam sebuah *chain* (-L)
5. membersihkan aturan di luar sebuah *chain* (-F)
6. menjadikan "nol" penghitungan paket dan byte pada semua aturan dalam sebuah *chain* (-Z)

Beberapa cara untuk memanipulasi aturan dalam sebuah *chain* yaitu :

1. Menambahkan sebuah aturan ke sebuah *chain* (-A).
2. Menyisipkan sebuah aturan pada posisi tertentu dalam suatu *chain* (-l).
3. Menggantikan aturan pada posisi tertentu dalam sebuah *chain* (-R).
4. Menghapus sebuah aturan pada posisi tertentu dalam suatu *chain* (-D).
5. Menghapus aturan pertama yang sesuai dalam sebuah *chain* (-D).