BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada Penelitian Rani dan Rao (2011) yang berjudul "*A Novel Implementation of ARM based Design of Firewall to prevent SYN Flood Attack*", Dalam paper ini mencoba mencegah jenis serangan dengan menggunakan *Iptables firewall* pada *ARM board*. Setelah mengimplementasikan *firewall*, eksperimen dilakukan untuk mempelajari sejauh mana *firewall* dapat mencegah serangan dan mengamankan terutama serangan *SYN Flood*. Hasil dari analisis menyatakan jika serangan *SYN Flood* terjadi di *ARM board*, maka *webserver* tidak dapat diakses, dan setelah aturan *Iptables* diimplementasikan ke dalam *ARM board*, jika penyerang akan mengirimkan serangan untuk *ARM board*, maka *webserver* bisa diakses karena *firewall Iptables* telah ditanamkan pada *ARM board*.

Pradana, Henryranu, dan Setiawan (2014), dalam penelitiannya yang berjudul "Desain dan Implementasi Keamanan Jaringan *Firewall* pada *Embedded System*" melakukan analisa *embedded firewall system* dengan simulasi serangan *scanning port*, *DoS* (*SYN flood*, *ICMP flood*, dan *UDP flood*), terhadap *resource CPU usage*, *RAM usage*, *network*, dan akses *webserver* dengan menggunakan *Linux Iptables*. Pengujian *embedded firewall system* ini bertujuan untuk mengetahui kemampuan *firewall* terhadap parameter serangan yang telah ditentukan. Pada saat pengujian simulasi serangan *scanning port* menggunakan *firewall*, hasil yang diperoleh adalah komputer penyerang tidak bisa mengetahui

nomor *port* yang terbuka dari komputer target. Pada pengujian simulasi serangan *SYN flood, ICMP flood,* dan *UDP flood* tidak menggunakan *firewall,* akses *webserver* terjadi kegagalan ketika terjadinya simulasi serangan. Pengujian ketika simulasi serangan dengan *firewall* menggunakan *embedded firewall system*, simulasi serangan *port scanning* dapat dilakukan filtering oleh *embedded firewall system* sehingga komputer penyerang tidak bisa mengetahui informasi nomor *port* dari komputer target. Pada hasil pengujian tersebut diperoleh kesimpulan bahwa *Linux Iptables* dapat diimplementasikan sebagai keamanan jaringan untuk mencegah serangan *scanning port* dan *Denial of Service* (*DoS*) (*SYN flood, ICMP flood, UDP flood*) pada *embedded system,* akses *webserver* berhasil dilakukan ketika simulasi serangan *ICMP flood* dengan paket perdetik 1.000 dan 10.000 paket. Simulasi serangan *SYN flood* dan *UDP flood* dengan paket 1.000 per detik.

Menurut Qasim (2012), dalam penelitiannya yang berjudul "*Mitigating DoS/ DDoS Attacks Using Iptables*", *Denial of Service* (*DoS*) adalah masalah keamanan jaringan yang merupakan tantangan serius untuk keandalan layanan yang ditujukan pada *server*. Tujuan dari serangan *DoS* adalah untuk menghabiskan sumber daya yang terdapat di sistem yang ditargetkan, mengurangi atau menghilangkan kemampuan layanan yang disediakan. Qasim (2012) membahas efisiensi teknik *packet filtering* dengan menggunakan *firewall* untuk tetap bertahan melawan serangan dari *DoS*. *Firewall* yang dimaksud menggunakan *Iptables* dalam *Linux* untuk menyangkal lalu lintas yang mencurigakan. Pada penelitian ini, kemampuan aturan *Iptables* di eksplorasi untuk bertahan terhadap serangan ini, untuk menentukan apakah lalu lintas jaringan sah

atau tidak, *Iptables* bergantung pada satu set aturan mengandung yang telah ditetapkan oleh jaringan atau *administrator system* sehingga aturan ini memberitahukan *Iptables* untuk mempertimbangkan dan apa yang harus dilakukan terhadap jaringan lalu lintas yang datang dari sumber tertentu, pergi ke tujuan tertentu, atau memiliki tipe protokol tertentu. Dasar analisis *script* menggunakan *Iptables* telah dikembangkan untuk memungkinkan atau menyangkal lalu lintas jaringan tergantung pada tingkat lalu lintas dari *Ip address* dari komputer pengirim paket.

2.2 Keamanan Jaringan Komputer

Keamanan komputer seperti yang dikatakan oleh Howard, seorang *Analysis Of Security Incidents On The Internet* pada tahun 1989-1995, mengatakan bahwa "C*omputer Security is preventing attackers from achieving objectives through unathorized access or unauthorized use of computers & Networks"* yaitu proses pencegahan yang dilakukan terhadap penyerang untuk terhubung kedalam jaringan komputer melalui akses yang tidak sah, atau penggunaan secara *illegal* dari komputer dan jaringan (Dewannanta, 2013).

Menurut Comer (2008: 607), keamanan jaringan dapat digambarkan secara umum yaitu segala sesuatu yang di buat untuk mengamankan jaringan. Dengan adanya keamanan jaringan diharapkan keamanan, integritas, kehandalan dari jaringan secara umum dan data yang ada di dalamnya menjadi lebih baik dan aman.

Di dalam keamanan jaringan terdapat pula risiko jaringan komputer yang merupakan segala bentuk ancaman baik fisik maupun *logic* yang langsung atau

tidak langsung mengganggu kegiatan yang sedang berlangsung dalam jaringan. Resiko dalam jaringan komputer disebabkan oleh beberapa faktor yaitu:

a. Kelemahan manusia;
b. Kelemahan perangkat keras computer;
c. Kelemahan sistem operasi jaringan;
d. Kelemahan sistem jaringan komunikasi.

Comer (2008: 608) juga mengemukakan bahwa keamanan jaringan mempunyai beberapa tujuan yang dapat membuat keamanan jaringan lebih baik lagi, yaitu:

a. Confidentiality

Prinsip pengamanan informasi yang menjamin terjaganya informasi dari kebocoran di mana akses terhadap data dan informasi hanya dapat dilakukan oleh pihak yang memiliki otorisasi.

b. Integrity

Prinsip pengamanan informasi yang menjamin keutuhan dan keaslian data dan informasi, di mana data dan informasi tidak berubah tanpa izin dari pihak yang memiliki otorisasi dengan tujuan untuk memastikan keakuratan dan kelengkapan informasi tersebut.

c. Availability

Prinsip pengamanan informasi yang menjamin diperolehnya informasi yang benar dan dapat diakses bila dibutuhkan oleh pihak yang memiliki otorisasi.

Tujuan keamanan jaringan dapat dicapai dengan suatu metode keamanan jaringan yang dapat melindungi sistem baik dari dalam maupun dari luar jaringan, namun bukan hanya melindugi tetapi dapat bertindak apabila terjadi serangan yang ada didalam jaringan. Salah satu metode tersebut yaitu *Iptables.* Namun, selain metode tersebut dibutuhkan juga suatu pemahaman tentang bagaimana menentukan kebijakan keamanan *(security policy)* dalam keamanan jaringan. Jika ingin menentukan apa saja yang harus dilindungi maka harus mempunyai perencanaan keamanan yang matang dan baik berdasarkan pada prosedur dan kebijakan keamanan jaringan, apabila tidak direncanakan maka tidak akan sesuai dengan yang diharapkan dalam perlindungan jaringan.

2.3 Kebijakan Keamanan

Salah satu masalah *network security* yang paling penting adalah menentukan kebijakan dalam *network security*. Kebanyakan orang menginginkan solusi teknis untuk setiap masalah yaitu dapat berupa program yang dapat memperbaiki masalah-masalah *network security*. Padahal, perencanaan keamanan yang matang berdasarkan prosedur dan kebijakan dalam *network security* akan membantu menentukan apa-apa yang harus dilindungi, berapa besar biaya yang harus ditanamkan dalam melindunginya, dan siapa yang bertanggung jawab untuk menjalankan langkah langkah yang diperlukan untuk melindungi bagian tersebut. Di dalam keamanan jaringan, peran manusia memegang tanggung jawab keamanan yang cukup berperan. Keamanan jaringan tidak akan efektif kecuali orang-orangnya mengetahui tanggung jawabnya masing-masing. Dalam menentukan *network security policy*, diperlukan adanya ketegasan apa yang

diharapkan, serta dari siapa hal tersebut diharapkan. Selain itu, kebijakan ini harus mencakup:

1. Tanggung jawab keamanan *network user*, meliputi antara lain keharusan user untuk mengganti *password*-nya dalam periode tertentu, dengan aturan tertentu, atau memeriksa kemungkinan terjadinya pengaksesan oleh orang lain.
2. Penggunaan yang benar sumber-sumber *network*, dengan menentukan siapa yang dapat menggunakan sumber-sumber tersebut, apa yang dapat dan tidak boleh dilakukan.
3. Langkah-langkah yang harus diperbuat bila terdeteksi masalah keamanan, siapa yang harus diberitahu. Hal ini harus dijelaskan dengan lengkap, bahkan hal-hal yang sederhana seperti menyuruh *user* untuk tidak mencoba melakukan apa-apa atau mengatasi sendiri bila masalah terjadi, dan segera memberitahu sistem administrator.

Adanya kebijakan tersebut maka manusia merupakan salah satu faktor yang sangat penting, namun sering dilupakan dalam pengembangan teknologi informasi, begitu juga dengan pengembangan di bidang keamanan jaringan. Salah satu contohnya adalah dalam penggunaan *password* yang sulit justru menyebabkan pengguna menuliskannya pada kertas yang ditempelkan pada komputer atau meja. Kebijakan keamanan dapat dijaga atau disusun karena faktor manusia dan budaya setempat juga harus diperhitungkan dan dipertimbangkan. Selain faktor dari manusia dan budaya itu sendiri, faktor yang dibutuhkan juga

tergantung dari organisasi, keputusan yang di ambil merupakan keputusan tentang keamanan komputer, dan juga masalah biaya dari suatu sistem keamanan.

2.4 *Webserver*

Menurut Panduan Keamanan *Webserver* (2011: 16), *Webserver* merupakan suatu komputer yang menyediakan layanan *World Wide Web* (*WWW*) pada *internet*, yang mencakup perangkat keras, sistem operasi, perangkat lunak *webserver*, dan konten situs *website* (halaman *web*). Jika *webserver* tersebut digunakan secara internal dan tidak oleh publik, maka dikenal sebagai suatu "*server* intranet".

Apabila sebuah *web browser* berhubungan dengan suatu situs *web*, sebetulnya ia berhubungan dengan sebuah *webserver*. *Server* tersebut mendengarkan (*listen*) *request* pada jaringan dan memberikan jawaban berupa data tertentu kepada client atau pengirim permintaan. *Webserver* atau *HTTP* adalah sebuah program yang melayani koneksi *HTTP* (*Hyper Text Transfer Protocol*) yang bekerja di *port* 80 (secara *default*, namun dapat diubah sesuai keinginan). *Webserver* bekerja berdasarkan *request-response*, yaitu ketika *HTTP client* (*web browser*) membangun koneksi dengan mengirimkan *request* (permintaan) kepada *server*, maka *server* akan merespon dengan mengelola permintaan tersebut kemudian mengirimkan data sesuai yang diminta oleh *HTTP Client*. *HTTP* merupakan protocol yang bekerja pada lapisan aplikasi (*application layer*) dan secara sederhana dapat didefinisikan sebagai sekumpulan aturan untuk tukar menukar data pada *World Wide Web* (*WWW*) (Rohman, 2010).

Ardian, Rochim, dan Widianto (2013: 30), menyatakan bahwa *server web* dapat merujuk baik pada perangkat keras ataupun perangkat lunak. *Server web* menyediakan layanan akses kepada pengguna melalui protokol komunikasi *HTTP* atau *HTTPS*. Penggunaan paling umum *server web* adalah untuk menempatkan situs *web*. Pada prakteknya penggunaannya diperluas sebagai tempat peyimpanan data ataupun untuk menjalankan sejumlah aplikasi kelas bisnis. Fungsi utama sebuah *server web* untuk mentransfer berkas atas permintaan pengguna melalui protokol komunikasi yang telah ditentukan dan mentransfer seluruh aspek pemberkasan dalam sebuah halaman *web* yang terkait; termasuk di dalamnya teks, gambar, video, atau lainnya. Kriteria dasar *webserver* berjalan dengan baik adalah dengan terjalinnya komunikasi misalnya protokol *HTTP* atau *HTTPS* dari *server* ke klien atau sebaliknya tanpa ada data yang hilang.

Webserver seringkali merupakan *host* yang paling banyak menjadi sasaran dan diserang di lingkup jaringan suatu organisasi. Akibatnya, mengamankan *webserver* dan infrastruktur jaringan yang mendukungnya merupakan hal yang perlu dilakukan. Ancaman keamanan secara spesifik terhadap *webserver* umumnya terbagi menjadi beberapa kategori:

1. Entitas *malicious* dapat mengeksploitasi *bug* perangkat lunak dalam *webserver*, sistem operasi *webserver*, atau *active content* untuk memperoleh akses ilegal ke dalam *webserver*.
2. Serangan *Denial of Service* (*DoS*) yang diarahkan pada *webserver* atau infrastruktur jaringan pendukungnya, sehingga dapat menolak atau menghalangi para pengguna sah yang akan memanfaatkan layanannya.

3. Informasi sensitif pada *webserver* yang memungkinkan untuk dibaca atau dimodifikasi tanpa otorisasi.
4. Informasi sensitif di *database* penyangga yang digunakan untuk mendukung elemen interaktif dari suatu aplikasi *web* memungkinkan untuk dibobol melalui serangan *command injection*, misalnya injeksi *Structured Query Language* [SQL], injeksi *Lightweight Directory Access Protocol* [LDAP], *cross-site scripting* [XSS].
5. Informasi sensitif yang ditransmisikan tanpa dienkripsi antara *webserver* dan *browser* yang dapat disadap dan terbaca dengan jelas.
6. Informasi pada *webserver* yang dapat diubah untuk tujuan jahat. *Defacement* (penggantian tampilan) situs *web* merupakan contoh yang umumnya dilaporkan untuk ancaman ini.
7. Entitas *malicious* yang berhasil mendapatkan akses illegal terhadap sumber daya di tempat lain dalam jaringan organisasi melalui suatu serangan terhadap *webserver*.
8. Entitas *malicious* yang menyerang organisasi internal setelah membobol suatu *host webserver*. Serangan tersebut dapat dilancarkan secara langsung, misalnya dari *host* yang bobol terhadap suatu *server* internal atau secara tidak langsung misalnya dengan menempatkan konten *malicious* pada *webserver* yang bobol yang berusaha mengeksploitasi kerawanan dalam *browser web* dari para pengguna yang mengunjungi situs tersebut.

9. *Server* yang dapat digunakan sebagai suatu titik distribusi perangkat serangan, pornografi, atau lunak yang di-*copy* secara ilegal.

Tipe serangan tidak langsung terhadap *webserver* bertujuan mendapatkan informasi dari para penggunanya. Sebagai contoh, pengguna dibujuk atau secara otomatis diarahkan untuk mengunjungi situs *web malicious* tampak tidak mencurigakan (karena menyerupai *website* aslinya). Melalui aktivitas yang dilakukan pengguna pada *website malicious* tersebut, di dapatkan informasi penting pengguna sepeti *username, password*, dll.. Hasil perolehan tersebut dapat disalahgunakan untuk mengakses *website* aslinya secara sah atau dengan kata lain merupakan salah satu upaya pencurian identitas. Serangan yang berhasil mampu membobol rahasia sumber daya situs *web* atau merusak gambaran atau profil mengenai organisasi. Serangan tidak langsung ini dapat berbentuk:

1. *Phishing*, dimana para penyerang menggunakan *social engineering* (rekayasa sosial) untuk memperdaya para pengguna agar melakukan *logging* ke suatu situs palsu.
2. *Pharming*, dimana *server DNS* atau *file host* para pengguna dibobol sehingga para pengguna diarahkan ke suatu situs *malicious* pengganti situs yang sah.

Para administrator *webserver* merupakan arsitek sistem yang bertanggungjawab terhadap keseluruhan rancangan, implementasi, dan pemeliharaan suatu *webserver*. Para administrator jaringan bertanggungjawab terhadap keseluruhan desain, implementasi, dan pemeliharaan suatu jaringan.

Administrator bertanggungjawab terhadap aktivitas yang berkaitan dengan *webserver* berikut:

a. Meng-*install* dan mengkonfigurasi sistem dalam kerangka pemenuhan kebijakan keamanan organisasi dan sistem standar dan konfigurasi jaringan.
b. Memelihara sistem dengan cara yang aman, termasuk *backup* berkali-kali dan aplikasi *patch* tepat waktu.
c. Memonitor integritas sistem, tingkat perlindungan, dan kejadian-kejadian yang berkaitan dengan keamanan.
d. Menindaklanjuti gejala *anomaly* keamanan yang terdeteksi yang muncul bersamaan dengan sumber daya sistem informasinya.
e. Melakukan pengujian keamanan sebagaimana yang dibutuhkan.

2.4.1 Keamanan *Web*

Keamanan *web* adalah suatu hal yang sangat penting dalam *web* karena *web* memiliki konten yang harus dilindungi. Penerapan keamanan *web* adalah berupa pencegahan akses kedalam *web* oleh pengguna yang tidak dikenal. Siapaun dapat menembus masuk kedalam *web* dan dapat memperoleh data-data yang tersimpan didalam *web*. Oleh karena itu, dalam mengamankan suatu *web*, suatu keamanan harus memiliki standard aspek dasar keamanan yang sudah teruji. Berikut ini akan dibahas mengenai aspek dasar keamanan jaringan dan *web*.

2.4.1.1 Aspek Dasar Keamanan *Web*

Secara umum terdapat 5 aspek dasar keamanan aplikasi berbasis *web* yang harus diperhatikan, yaitu otentikasi, otorisasi, integritas data, non *repudiation*, dan kontrol akses.

1. Otentikasi

   Otentikasi merupakan proses untuk mengidentifikasi pengirim ataupun penerima. Otentikasi juga dapat disebut sebuah verifikasi apakah seseorang itu adalah orang yang berhak. Seseorang disini maksudnya adalah pengguna yang dapat mengakses aplikasi berbasis *web*. Otentikasi ini biasanya melibatkan sebuah identitas seperti contohnya adalah *username* dan kata sandi. Pengguna yang dapat mengakses aplikasi berbasis harus memiliki identitas yang sudah diotentikasi, maksudnya memiliki *username* dan kata sandi (Liyantanto, 2009).

2. Otorisasi

   Otorisasi adalah sebuah proses pencarian apakah pengguna yang identitasnya sudah diidentifikasi (otentikasi), diijinkan untuk memanipulasi sumber daya tertentu. Ini biasanya ditentukan dengan mencari apakah orang itu merupakan bagian dari aturan khusus yang memiliki akses ke sumber daya (Liyantanto, 2009).

3. Integritas Data

   Integritas data memastikan bahwa data atau informasi yang dikirimkan dari sumber ke tujuan (*server* ke *client*) selama pengiriman tidak mengalami perubahan sesuai dengan data atau informasi aslinya. Data

atau informasi selama pengiriman dapat saja diubah oleh *user* atau aplikasi yang tidak memiliki hak untuk melakukan perubahan data. Oleh karena itu, layanan integritas data sangat penting (Liyantanto, 2009).

4. Non-*Repudiation*

   Layanan ini menjamin bahwa *server* dan *client* yang berhubungan tidak dapat menyangkal hubungan yang terjadi. Sehingga seorang pengirim pesan dapat memastikan bahwa pesan yang dikirmnya benar-benar diterima oleh penerima (Liyantanto, 2009).

5. Kontrol Akses

   Suatu kemampuan untuk melakukan pembatasan akses ke sistem *host* dan aplikasinya, yang dilakukan melalui jalur-jalur komunikasi (Weippl, 2005).

2.5 *Penetration Testing*

Penetration testing adalah metode komprehensif untuk menguji kelengkapan, terintegrasi, operasional, dan dasar komputasi terpercaya yang terdiri dari perangkat keras, perangkat lunak dan orang. Proses tersebut melibatkan analisis sistem untuk setiap kerentanan potensial, termasuk yang buruk atau konfigurasi sistem yang tidak benar, kekurangan hardware dan software, serta kelemahan operasional dalam proses atau teknik penanggulangan. (Bacudio, Yuan, Chu, dan Jones, 2011)

Menurut Kennedy, O'Gorman, Kearns, dan Aharoni (2011) *penetration testing* adalah metode untuk mengevaluasi keamanan sistem komputer atau jaringan dengan mengsimulasikan serangan dari sumber yang berbahaya. Sebagai contoh serangan yang dilakukan oleh "*Black Hat Hacker*", "*Cracker*", "*Defacer*" dan sebagainya.

Proses ini melibatkan analisis aktif terhadap sistem untuk setiap kerentanan potensial yang diakibatkan oleh sistem yang lemah atau konfigurasi sistem yang tidak benar atau kelemahan operasional dalam proses teknis. Masalah keamanan yang ditemukan akan disampaikan kepada pemilik sistem bersama dengan penilaian dampak dan mitigasi (solusi teknis) dari setiap kerentanan yang ditemukan.

Tujuan dari *penetration testing* diantaranya adalah untuk menentukan dan mengetahui serangan-serangan yang bisa terjadi terhadap kerentanan yang ada pada sistem, mengetahui dampak bisnis yang diakibatkan dari hasil ekpoitasi yang dilakukan oleh penyerang. Sedangkan, *penetration tester* adalah orang yang melakukan *penetration testing* yaitu orang yang melakukan evaluasi keamanan sistem komputer atau jaringan dengan mengsimulasikan serangan dari sumber yang berhahaya.

2.6 *Denial of Service* (*DoS*)

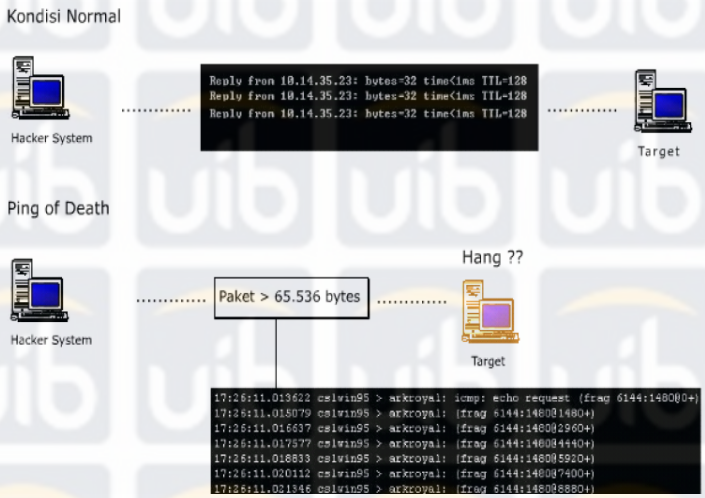
Denial Of Service (*DoS*) *attacks* adalah jenis serangan terhadap sebuah komputer atau *server* di dalam jaringan *internet* dengan cara menghabiskan sumber (*resource*) yang dimiliki oleh komputer tersebut sampai komputer tersebut tidak dapat menjalankan fungsinya dengan benar sehingga secara tidak langsung

mencegah pengguna lain untuk memperoleh akses layanan dari komputer yang diserang tersebut (Rafiudin, 2009).

Qasim (2012) mengemukakan berbagai macam-macam tipe serangan *DoS* dapat dilihat dari protokol jaringan yang akan digunakan, seperti *TCP, UDP, ICMP*. Beberapa tipe serangan *Denial of Service* antara lain *Ping of Death, SYN Attack, Land Attack, Smurf Attack, dan UDP Flood.* (Nurwenda, Irawan, dan Irzaman, 2004). Berikut ini adalah penjelasan masing-masing tipe serangan *Denial of Service* (*DoS*):

1. *Ping of Death*

Pada kondisi normal program *utility ping* digunakan untuk mengecek beberapa waktu yang dibutuhkan untuk mengirimkan sejumlah data dari suatu komputer ke komputer lain, dimana panjang maksimum paket data yang dapat dikirimkan menurut spesifikasi protokol IP adalah 65.536 byte. Gambar 2.1. menunjukkan program *utility ping* dan alur serangan *Ping of Death.* Pada *Ping of Death* data yang dikirim melebihi maksimum paket yang di izinkan menurut spesifikasi protokol *IP*. *Ping of Death* mengeksploitasi kelemahan didalam *reassembly* kembali fragmen paket *IP*. Ketika data dikirimkan ke jaringan, paket *IP* sering berubah menjadi potongan paket yang lebih kecil. *Ping of Death* akan memanipulasi *offset* potongan data sehingga akhirnya terjadi *overlapping* antara paket yang diterima di bagian penerima setelah potongan-potongan paket ini di *reassembly*. Konsekuensinya, pada sistem yang tidak siap akan menyebabkan sistem tersebut crash (tewas), hang (bengong) atau reboot (booting ulang) pada saat sistem tersebut menerima paket yang demikian panjang.

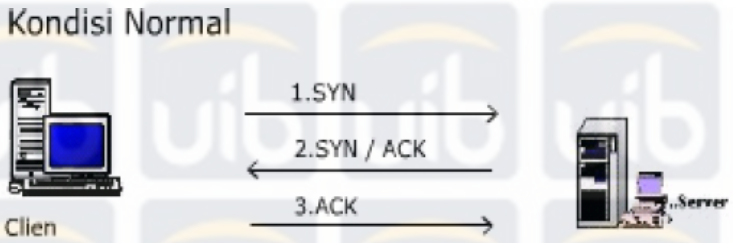


Gambar 2.1. Skema Program *Utility Ping* dan Serangan *Ping of Death*

Sumber: Nurwenda, Nurwenda, Irawan, dan Irzaman (2004)

2 Serangan *SYN Flood*

Pada kondisi normal, aplikasi klien akan mengirimkan paket *TCP SYN* untuk mensinkronisasi paket pada aplikasi di server (penerima). Server (penerima) akan mengirimkan respond acknowledgement berupa paket *TCP SYN -ACK*. Setelah paket *TCP SYN -ACK* di terima dengan baik oleh klien (pengirim), maka klien (pengirim) akan mengirimkan paket ACK sebagai tanda transaksi pengiriman atau penerimaan data akan di mulai. Proses ini disebut juga dengan *three-way handshake.* Hal ini terlihat pada gambar 2.2. berikut ini



Gambar 2.2. Skema Koneksi *TCP Three-way Handshake*

Sumber: Nurwenda, Nurwenda, Irawan, dan Irzaman (2004)

Gambar 2.3. Skema Serangan *SYN*

Sumber: Nurwenda, Nurwenda, Irawan, dan Irzaman (2004)

Dalam serangan *SYN flood* seperti terlihat pada gambar 2.3. diatas, *hacker* mengirimkan paket *SYN* yang *source address*-nya telah di-*spoof* menjadi suatu *address* yang tidak ada dan dikirimkan ke *server*. *Server* membalas dengan mengirimkan *SYN/ACK* ke alamat palsu (*spoof*), dan port yang digunakan berada pada kondisi SYN_RECV. Jika seandainya alamat *spoff* (palsu) tersebut ada (terdapat sistem atau komputer) maka akan membalas dengan paket RST karena merasa tidak memulai koneksi. Dikarenakan alamat palsu tersebut tidak ada maka koneksi yang telah dialokasikan pada *port server* tersebut akan berada pada keaadaan menunggu (75 detik – 23 menit). Dengan cara ini, *server* akan tampak seperti bengong dan tidak memproses *responds* dalam waktu yang lama.

3. *Land Attack*

Dalam *Land attack*, *hacker* menyerang *server* yang dituju dengan mengirimkan paket TCP SYN palsu yang seolah-olah berasal dari *server* yang dituju. Dengan kata lain, *source* dan *destination address* dari paket dibuat seakan-akan berasal dari *server* yang dituju. Akibatnya, *server* yang diserang menjadi bingung. Apabila serangan diarahkan kepada sistem *windows* 95, maka sistem

yang tidak diproteksi akan menjadi hang (dan bisa keluar layar biru). Hal ini terlihat pada gambar 2.4 berikut ini.

Gambar 2.4. Skema Serangan *Land*

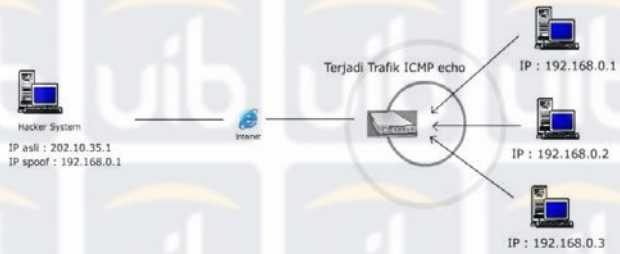
Sumber: Nurwenda, Nurwenda, Irawan, dan Irzaman (2004)

4. *Smurf Attack*

Pada *smurf attack*, *hacker* membanjiri router dengan paket permintaan echo *Internet Control Message Protocol* (*ICMP*) yang di kenal sebagai aplikasi *ping*, karena alamat *IP* tujuan pada paket yang dikirim adalah alamat *broadcast* dari jaringan, maka *router* akan mengirimkan permintaan *ICMP echo* ini ke semua mesin yang ada di jaringan. Apabila ada banyak *host* di jaringan, maka akan terjadi trafik *ICMP echo respons* dan permintaan dalam jumlah yang sangat besar.

Pada gambar 2.5, *hacker* memilih untuk men-*spoof* alamat *IP* sumber permintaan *ICMP* tersebut, Dengan menggunakan *IP spoofing*, respon dari *ping* tadi dialamatkan ke komputer yang *IP*-nya di-*spoof* sehingga komputer tersebut akan menerima banyak paket. Akibatnya terjadi ICMP trafik yang tidak hanya akan memacetkan jaringan komputer perantara saja, tapi jaringan yang alamat IP-nya di spoof.

ICMP Flood memanfaatkan *protokol Internet Control Message Protocol (ICMP)* sebagai perantara penyerangan. *User* akan mengirimkan paket *echo* ke remote *host* untuk mengecek apakah *server* masih aktif. Selama serangan *ICMP Flood DoS*, agen akan mengirim *ICMP_ECHO_REPLY (PING)* dalam volume yang besar kepada korban atau *server*.

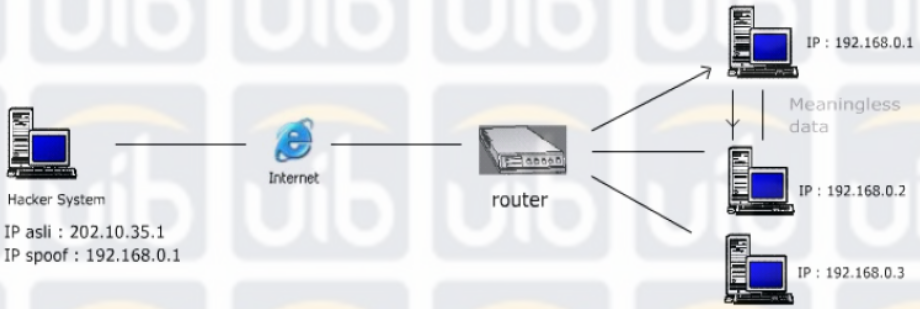


Gambar 2.5. Skema Serangan *Smurf Attack*

Sumber: Nurwenda, Nurwenda, Irawan, dan Irzaman (2004)

5. Serangan *UDP Flood*

Pada *UDP Flood*, penyerang mengirimkan paket *UDP* dalam jumlah besar ke *sistem* korban, sehingga muncul *saturation* jaringan dan berkurangnya *bandwith* yang tersedia untuk melayani permintaan layanan yang sah untuk sistem korban. Pada *UDP attack* dengan cara spoofing, *user datagram protocol* (*UDP*) *flood attack* akan menempel pada servis *UDP chargen* di salah satu mesin, yang akan mengirimkan sekelompok karakter ke mesin lain, yang di program untuk meng-echo setiap kiriman karakter yang di terima melalui servis *chargen*. Hal ini ditunjukkan oleh gambar 2.6. dimana *UDP Flood* ini pada dasarnya mengkaitkan dua (2) sistem tanpa disadarinya. Karena paket *UDP* tersebut di *spoofing* antara ke dua mesin tersebut, maka yang terjadi adalah banjir tanpa henti kiriman karakter yang tidak berguna antara ke dua mesin tersebut.



Gambar 2.6. Skema Serangan *UDP*

Sumber: Nurwenda, Nurwenda, Irawan, dan Irzaman (2004)

2.7 *Internet*

Internet merupakan singkatan dari *Interconnection Networking*. *Internet* berasal dari bahasa latin "*inter*" yang berarti antara. Secara kara perkata *INTERNET* berarti jaringan antara atau penghubung, sehingga kesimpulan dari definisi *internet* merupakan hubungan antara berbagai jenis komputer dan jaringan di dunia yang berbeda sistem operasi maupun aplikasinya dimana hubungan tersebut memanfaatkan kemajuan komunikasi (telepon dan satelit) yang menggunakan protokol standar dalam berkomunikasi yaitu protokol *TCP/IP* (*Tranmission Control*/*Internet Protocol*) (Supriyanto dan Muhsin, 2008).

2.8 *OSI Layer*

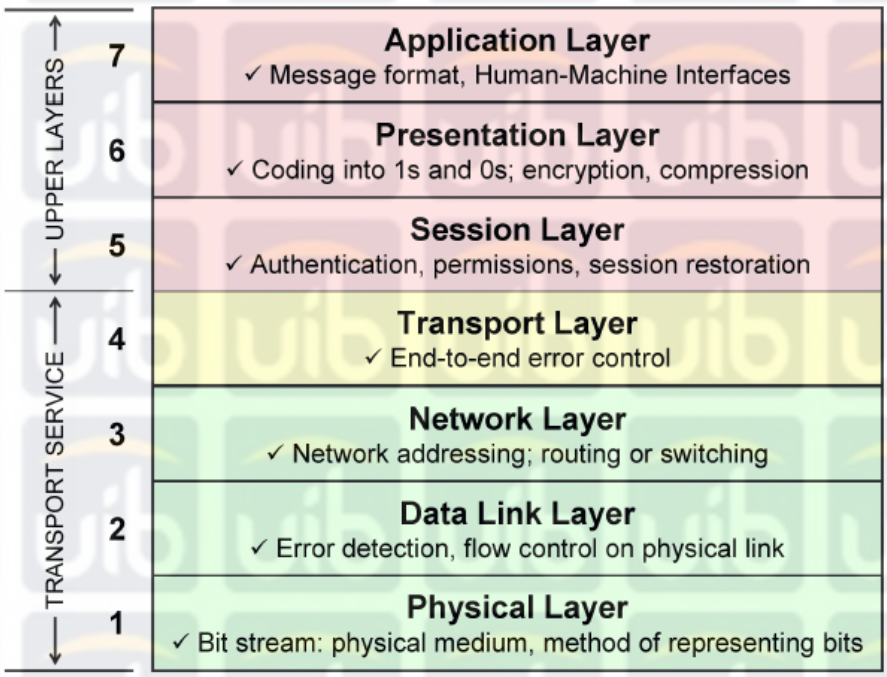
Menurut Tanenbaum (2011: 41), *OSI* adalah singkatan dari *Open System Interconnection*, diperkenalkan pada tahun 1984 oleh *ISO* (*International Organization for Standardization)*. *OSI* model pada dasarnya dikembangkan untuk menyederhanakan jaringan yang kompleks, memfasilitasi pelatihan jaringan, dan membuat *troubleshooting* jaringan menjadi lebih mudah.

OSI model adalah suatu standar antar mesin yang terdiri atas 7 lapisan yang memiliki fungsi yang berbeda**.** Ketujuh lapisan tersebut mempunyai peran dan fungsi yang berbeda satu terhadap yang lain. Setiap *Layer* bertanggung jawab secara khusus pada proses komunikasi data. Misalnya, satu *layer* bertanggung jawab untuk membentuk koneksi antar perangkat, sementara *layer* lainnya bertanggung jawab untuk mengoreksi terjadinya *error* selama proses transfer data berlangsung. Model *Layer OSI* dibagi dalam dua group, yaitu *upper layer* dan *lower layer*. *Upper layer* fokus pada aplikasi pengguna dan bagaimana *file* direpresentasikan di komputer. Untuk *Network Engineer*, bagian utama yang menjadi perhatian adalah pada *lower layer*. *Lower layer* adalah intisari komunikasi data melalui jaringan *actual* (Sopandi, 2008).

Tanenbaum (2011: 43) menjelaskan *layer* beserta fungsi yang terdapat dalam *OSI*, yaitu:

1. *Physical Layer*. *Layer* ini mendefinisikan fungsi *logic levels, data rate, physical media,* dan *data conversion* yang menyusun paket dari satu perangkat ke perangkat lainnya
2. *Data Link Layer. Layer* ini memproses paket yang lewat dan membuka data di dalam paket.
3. *Network Layer. Layer* ini menyediakan alamat *routing* untuk setiap paket yang dikirimkan melalui *logical addressing* dan fungsi *switching*
4. *Transport Later. Layer* ini menyediakan fungsi *QoS* dan memastikan bahwa paket telah terkirim dengan sempurna.

5. *Session Layer*. *Layer* ini mengelola koneksi antara 2 perangkat, membuat komunikasi, menjaga komunikasi, dan memutuskan komunikasi yang sudah dibuat.
6. *Presentation Layer*. *Layer* ini mengecek data untuk memastikan apakah *compatible* dengan sumber daya komunikasi yang ada. *Layer* ini juga mengelola *data formatting*, *data compression*, dan *encryption*.
7. *Application Layer*. *Layer* ini bekerja dengan *software* aplikasi untuk menyediakan fungsi komunikasi yang dibutuhkan. *Layer* ini juga bekerja dengan *domain name service* (*DNS*), *file transfer protocol* (*FTP*), *hypertext transfer protocol* (*HTTP*), *internet message access protocol* (*IMAP*), *post office protocol* (*POP*), dan *simple mail transfer protocol* (*SMTP*).



Gambar 2.7 *OSI Layer*

Sumber: Tanenbaum (2011: 43)

2.9 *Transmission Control Protocol* (*TCP*) & *User Datagram Protocol* (*UDP*)

Menurut Fadia (2006: 451) *Transmission Control Protocol* (TCP) adalah protokol di *transport layer* yang bersifat *connection-oriented*. *Transmission Control Protocol* (*TCP*) menjamin paket dikirimkan pada tujuan tanpa *error* dan data yang diterima urutannya sesuai dengan data yang dikirimkan. *Transmission Control Protocol* (*TCP*) memiliki beberapa fungsi:

- *Data transfer*: *TCP* memiliki kemampuan untuk mengirimkan arus data secara terus menerus antara 2 *user* melalui jaringan yang ada;
- *Reliable delivery*: *TCP* memiliki kemampuan untuk memperbaiki data yang rusak, hilang, dan telah terduplikasi di jaringan;
- *Flow control*: *TCP* menyediakan mekanisme yang membantu penerima untuk mengontrol jumlah data yang dapat dikirim oleh pengirim;
- *Multiplexing*: *TCP* menyediakan kumpulan *port* untuk setiap *host* sehingga setiap *host* dapat menggunakan fasilitas komunikasi *TCP* secara terus menerus.

Sedangkan, *User Datagram Protocol* adalah sebuah protokol sederhana di *transport layer*, dimana protokol *UDP* tidak menjamin keandalan dari paket yang dikirimkan. protokol *UDP* cocok digunakan pada saat paket pengiriman tepat pada waktunya lebih penting daripada paket tersampaikan atau tidak. Cara kerja protokol *UDP* sangat sederhana, protokol *UDP* melakukan enkapsulasi *data* dari *user* menjadi sebuah datagram. Kemudian *datagram* tersebut dikirim ke *IP layer* untuk proses transmisi (Fadia, 2006: 453).

2.10 *Firewall*

Rani dan Rao (2011) mendefinisikan *firewall* sebagai perangkat lunak dari sistem komputer atau jaringan yang dirancang untuk memblokir akses yang tidak sah pada saat melakukan komunikasi resmi. Ini adalah perintah atau set perintah konfigurasi untuk mengizinkan, menolak, mengenkripsi, mendekripsi, atau *proxy* semua (masuk dan keluar) lalu lintas komputer antara keamanan domain yang berbeda berdasarkan seperangkat aturan dan kriteria lainnya. *Firewall* adalah alat khusus, atau perangkat lunak yang berjalan pada komputer, yang memeriksa jaringan lalu lintas yang lewat melalui itu, dan menyangkal atau izin berdasarkan seperangkat aturan.

Firewall adalah perangkat atau sebuah program yang berfungsi untuk mengontrol aliran lalu lintas jaringan antar jaringan atau *host* yang menerapkan keamanan yang berbeda. *Firewall* sering dibahas dalam sebuah konteks konektivitas *internet*, tetapi *firewall* juga memiliki penerapan pada konektivitas jaringan lainnya. Dapat diambil sebuah contoh, banyak jaringan dalam perusahaan menggunakan atau mengimplementasikan *firewall* untuk membatasi konektivitas menuju dan dari suatu jaringaan internal yang digunakan untuk melayani fungsi yang lebih *spesifik* seperti personalia dan akuntansi. Dengan menggunakan *firewall* untuk mengontrol *konektivitas* masing-masing bagian dalam suatu jaringan untuk mecegah akses tidak sah ke dalam sistem. Penggunaan *firewall* yang tepat dapat memberikan keamanan jaringan yang baik dalam suatu jaringan maupun sebuah *host* (Scarfone dan Hotman , 2009).

2.10.1 Fungsi *Firewall*

a. Analisa dan *filter* paket

Data yang dikomunikasikan lewat protokol di *internet*, dibagi atas paket-paket. *Firewall* dapat menganalisa paket ini, kemudian memperlakukannya sesuai kondisi tertentu.

b. *Blocking* isi dan protokol

Firewall dapat melakukan *blocking* terhadap isi paket yang keluar dan masuk ke jaringan tersebut.

c. Autentikasi koneksi

Firewall umumnya memiliki kemampuan untuk menjalankan autentikasi identitas *user*, integritas dari satu *session*, dan melapisi transfer data pada jaringan.

2.11 *Iptables*

Iptables adalah *software* atau aplikasi pada sistem operasi *Linux* yang memungkinkan administrator jaringan untuk mengkonfigurasi tabel yang disediakan oleh kernel *Linux firewall*. *Iptables* digunakan untuk mengatur, memelihara, dan memeriksa tabel aturan *packet filtering IP* pada kernel *Linux*. Setiap tabel berisi sejumlah *rule* dan *chains*. (Pradana, Barlian, dan Setiawan, 2014)

2.11.1 Beberapa Macam *Packet Filtering Iptables*

1. INPUT

Mengatur paket data yang memasuki *firewall* dari arah intranet maupun *internet*. kita bisa mengelola komputer mana saja yang bisa mengakses *firewall*. misalnya: hanya komputer IP 192.168.1.xxx yang bisa SSH ke *firewall* dan yang lain tidak boleh**.**

2. OUTPUT

Mengatur paket data yang keluar dari *firewall* ke arah intranet maupun *internet*. Biasanya output tidak diset, karena bisa membatasi kemampuan *firewall* itu sendiri.

3. FORWARD

Mengatur paket data yang melintasi *firewall* dari arah *internet* ke intranet maupun sebaliknya. *Policy forward* paling banyak dipakai saat ini untuk mengatur koneksi *internet* berdasarkan *port, mac address* dan alamat *IP*. Selain aturan (*policy*) *firewall Iptables* juga mempunyai parameter yang disebut dengan TARGET, yaitu status yang menentukkan koneksi pada *Iptables* diizinkan lewat atau tidak. TARGET memiliki sejumlah keputusan untuk diterapkan terhadap suatu paket yang diawali dengan –j [jump]. TARGET terdapat tiga macam yaitu:

1) ACCEPT

Akses diterima. Apabila ditemukan paket yang sesuai dengan aturan untuk di-*ACCEPT*, maka *firewall* akan langsung menerima untuk kemudian meneruskan paket tersebut.

2) *REJECT*

Akses ditolak. Apabila ditemukan paket yang sesuai dengan aturan untuk di-*REJECT*, maka *firewall* akan langsung membuang paket tersebut namun disertai dengan mengirimkan pesan ERROR ICMP "*port unreachable*". Koneksi dari komputer klien yang melewati *firewall* langsung terputus, Biasanya terdapat pesan "*connection refused*". Target *reject* tidak menghabiskan *bandwidth internet* karena akses langsung ditolak, hal ini berbeda dengan *DROP*.

3) *DROP*

Apabila ditemukan paket yang sesuai dengan aturan untuk di-*DROP*, maka *firewall* akan langsung membuang paket tersebut tanpa mengirimkan pesan *error* apapun ke pengirim. Akses diterima tetapi paket data langsung dibuat oleh kernel, sehingga pengguna tidak mengetahui kalau koneksinya dibatasi oleh *firewall*, pengguna melihat seakan-akan *server* yang dihubungi mengalami permasalahan teknis. Pada koneksi *internet* yang sibuk dengan trafik tinggi target *drop* sebaiknya jangan digunakan.

2.11.2 Berbagai Macam Perintah *Iptables*

Berikut adalah sintaks perintah *Iptables*:

***Iptables* [*tipe-perintah*] [*chain*] [*tipe-parameter*] –j [*target*]**

A. Tipe Perintah

Tipe perintah yang bisa digunakan adalah sebagai berikut:

- **–L [*list*]**

  Perintah ini digunakan untuk menampilkan semua aturan yang telah dibuat sebelumnya

- **–A [*append*]**

  Perintah ini digunakan untuk menerapkan satu aturan baru yang akan ditempatkan di baris yang paling bawah dari aturan-aturan yang telah dibuat sebelumnya.

- **–I [*insert*]**

  Perintah ini digunakan untuk memasukkan aturan baru sekaligus menempatkan aturan tersebut pada baris yang kita tentukan sendiri.

- **–R [*replace*]**

  Perintah ini digunakan untuk memasukkan aturan baru yang diletakkan pada baris yang kita tentukan sendiri dan aturan yang ada pada baris tersebut akan dihapus.

- **–D [*delete*]**

  Perintah ini digunakan untuk menghapus baris aturan yang telah dibuat sebelumnya. Gunakan perintah *Iptables* –L terlebih dahulu untuk mengetahui urutan baris aturan yang ada.

- **–F[*flush*]**

  Perintah ini digunakan untuk menghapus semua aturan yang telah ditetapkan.

B. Tipe Paramater

Tipe parameter pada *Iptables* sangat bermanfaat untuk membuat sebuah aturan yang lebih spesifik lagi, misalnya berdasarkan *source destination, port, rate*. Adapun penjelasan masing – masing parameter adalah sebagai berikut:

- **–p [jenis protocol]**

  Parameter ini berungsi untuk membuat aturan berdasarkan jenis protokol yang digunakan, misalnya *TCP*, *UDP*, *ICMP*.

- **–d [alamat *IP* tujuan]**

  Parameter –d berfungsi untuk membuat aturan mengacu pada alamat *IP* tujuan dari paket yang dikirimkan.

- **–s [alamat *IP* sumber]**

  Parameter –s berfungsi untuk membuat aturan mengacu pada alamat *IP* asal paket yang dikirimkan.

- **--dport [alamat *port* tujuan]**

  Parameter –dport berfungsi untuk membuat aturan mengacu pada alamat *port* tujuan.