MEREDUKSI SERANGAN DENIAL OF SERVICES TERDISTRIBUSI PADA LINUX VIRTUAL SERVER MENGGUNAKAN HONEYPOT

Noven Indra Prasetya
Program Studi Teknik Informatika Fakultas Teknik
Universitas Wijaya Kusuma Surabaya
Jl. Dukuh Kupang XXV / 54, Surabaya, Jawa Timur
Email: noven@uwks.ac.id

Abstrak. *Daniel of Service atau DoS merupakan suatu serangan yang dapat menyebabkan satu atau beberapa komputer server tidak dapat melayani pengguna dikarenakan server tersebut down atau mati. Dengan kata lain serangan DoS menyebabkan komputer target akan mengalami kegagalan beroperasi yang mengakibatkan semua layanan tidak dapat diakses oleh pengguna. Banyak penelitian dilakukan untuk mengurangi dampak terhadap serangan DoS, namun metode penyerangan DoS juga mengalami perkembangan, yaitu penyerangan DoS dilakukan secara terdistribusi atau yang lebih dikenal dengan nama Distributed Daniel of Services atau DDoS. Dan serangan DoS yang dilakukan secara bersama-sama atau terdistribusi, memiliki dampak dan pencegahan beberapa kali lebih sulit untuk dicegah apabila dibandingkan dengan serangan DoS yang dilakukan secara tunggal. Linux Virtual Server (LVS) sebagai salah satu bentuk solusi peningkatan kinerja server pada penyedia layanan di internet dapat juga terkena serangan DoS atau DDoS, sehingga dibutuhkan sebuah sistem yang dapat digunakan untuk mereduksi dampak dari serangan tersebut dengan cara mengarahkan paket data yang berisi serangan DoS ke sebuah komputer palsu yang didalamnya berisi semua layanan server sesungguhnya, tujuannya supaya penyerang mengira kalau serangan yang dilakukan berhasil namun kenyataannya server yang sesungguhnya tetap aman dan layanan yang didalamnya tetap beroperasi. Pada penelitian ini, penulis mengusulkan suatu mekanisme mereduksi serangan DoS atau DDoS dengan cara mengarahkan paket data yang berisi serangan DoS ke Honeypot dengan tujuan supaya tidak mengganggu kinerja server yang sebenarnya. Dengan mekanisme yang diusulkan tersebut, diharapkan paket-paket yang terdeteksi sebagai serangan akan diarahkan oleh LVS director ke Honeypot dan gangguan yang terjadi pada server-server asli dapat direduksi dengan baik.*

Kata Kunci: Daniel of Services, Honeypot, Intrusion Detection System, DoS, DDoS

Dewasa ini, kejahatan di dunia internet atau yang dikenal dengan istilah *cybercrime* telah banyak menimbulkan kerugian yang sangat besar bagi perusahaan-perusahaan besar di dunia, bahkan telah diprediksi mencapai US$ 1 milion atas kerugian yang dideritanya. Perhitungan tersebut dilakukan oleh Lembaga Keamanan McAfee yang memonitor aksi penjahat *cyber* termasuk pelaku pencurian dan pembobolan data sepanjang tahun 2008 lalu (Junaidi, 2010).

Sebagai contoh dari sekian banyak kejahatan di dunia internet yang sangat meresahkan adalah kejahatan *Daniel of Services*. *Daniel of Service* atau DoS merupakan suatu serangan yang dapat menyebabkan satu atau beberapa komputer server tidak dapat melayani pengguna dikarenakan server tersebut *down* (Ramamohanara, 2007). Dengan kata lain serangan DoS menyebabkan komputer target akan mengalami kegagalan beroperasi yang mengakibatkan semua layanan tidak dapat diakses oleh pengguna.

Banyak penelitian dilakukan untuk mengurangi bahkan mencegah dampak yang ditimbulkan oleh serangan DoS, namun bukan hanya metode pencegahan terhadap serangan DoS saja yang berkembang, melainkan metode penyerangan DoS juga mengalami perkembangan, yaitu penyerangan DoS dilakukan secara terdistribusi atau yang lebih dikenal dengan nama *Distributed Daniel of Services* atau DDoS (Ramamohanara, 2007). DDoS dilakukan dengan cara menginfeksi beberapa komputer pengguna lain dengan *software* DoS, kemudian tanpa disadari komputer yang terinfeksi *software* DoS diperintahkan oleh penyerang untuk bersama-sama melakukan serangan DoS kepada komputer atau server target penyerangan. Serangan DoS yang dilakukan secara bersama-sama atau terdistribusi, memiliki dampak dan pencegahan beberapa kali lebih sulit untuk dicegah apabila dibandingkan dengan serangan

DoS yang dilakukan secara tunggal (Ramamohanara, 2007).

Penelitian ini dilakukan untuk mereduksi dampak serangan DDoS pada *Linux Virtual Server* atau LVS. Alasan peneliti menggunakan LVS sebagai lingkungan pengujian pada penelitian ini dikarenakan pesatnya perkembangan pengguna internet yang mengakibatkan penyedia layanan di internet harus terus meningkatkan layanannya dengan baik. Salah satu upaya penyedia layanan di internet dalam meningkatkan layanannya adalah dengan mengganti server lama dengan server baru yang lebih canggih. Namun solusi mengganti server cukup merepotkan oleh sebagian administrator jaringan dikarenakan membutuhkan biaya yang besar, dan dibutuhkan waktu yang lama untuk konfigurasi ulang server, yang menyebabkan layanan tersebut harus mati untuk beberapa saat hingga server hidup kembali seperti sedia kala.

Solusi lain dari penggantian server adalah dengan melakukan penambahan server. Dengan penambahan server, layanan yang sudah ada tetap berjalan ketika proses konfigurasi sedang dilakukan, sehingga tidak akan sampai mematikan layanan yang sedang berjalan. Penambahan server tentu harus dilakukan dalam waktu yang cepat dan sebisa mungkin tidak diketahui oleh user. Karena dengan menambah jumlah server, maka harus dipikirkan pula pengaturan ketika salah satu server mati. Diharapkan layanan harus tetap berjalan meskipun ada satu atau beberapa server yang mati.

Oleh karena itu dibutuhkan suatu teknologi yang mampu menangani permasalahan tersebut dengan baik, yaitu teknologi *Linux Virtual Server* (LVS) (Zhang, 2000). Dengan LVS, beberapa server dapat memiliki hanya 1 buah IP address saja, dan IP address itulah yang nantinya akan diakses oleh user. Ketika user mengakses salah satu layanan yang ada, maka LVS akan mengatur agar user tersebut mendapat layanan dari salah satu server yang ada. Apabila ternyata salah satu server ada yang mati, maka layanan yang ada pada server tersebut dapat dipindah ke server yang lain secara otomatis tanpa harus membuat layanan yang berjalan mati akibat perpindahan server.

Pada penelitian ini, penulis mengusulkan suatu mekanisme mereduksi serangan DoS atau DDoS dengan cara mengarahkan paket data yang berisi serangan DoS ke *Honeypot* dengan tujuan supaya tidak mengganggu kinerja server yang sebenarnya. Dengan mekanisme yang diusulkan tersebut, diharapkan paket-paket yang terdeteksi sebagai serangan akan diarahkan oleh *LVS director* ke *Honeypot* dan gangguan yang terjadi pada server-server asli dapat direduksi dengan baik.

Tujuan dari penelitian ini adalah untuk mereduksi dampak yang ditimbulkan terhadap serangan DoS dan DDoS pada *Linux Virtual Server*. Sehingga diharapkan administrator jaringan yang menggunakan *Linux Virtual Server* sebagai arsitektur server-nya dapat memberikan layanan yang lebih baik kepada para pengguna meskipun sedang diserang menggunakan DoS maupun DDoS.

Honeypot

Pada buku yang berjudul *Honeypots:Tracking Hackers* menyebutkan bahwa *Honeypot* merupakan suatu *security resource* atau sistem yang tidak memiliki nilai *production*, dan didesain sedemikian mungkin agar sistem tersebut bisa di-*probe*, diserang, atau di-*compromised* (Spitzner, 2003). Sedangkan definisi lain dari *Lance Spitzner* yang dikutip pada situs "www.tracking-hackers.com" menyebutkan bahwa suatu *Honeypot* merupakan sumber sistem informasi yang menghasilkan nilai palsu pada saat terjadi penggunaan sumber daya yang tidak sah atau tidak diijinkan.

a. Tujuan Menggunakan Honeypot

Untuk lebih mengetahui penggunaan *Honeypot*, dibawah ini akan dijelaskan beberapa alasan menggunakan *Honeypot* (Spitzner, 2003).

1. Early Detection
   Early Detection atau Pendeteksian Dini adalah suatu metode yang akan memberitahukan dan mengingatkan pengguna pada serangan-serangan terhadap sistem server oleh orang-orang yang tidak memiliki otoritas.
2. New Threat Detection
   New Threat Detection atau Pendeteksian Ancaman Baru adalah suatu metode yang digunakan untuk mengetahui ancaman-ancaman baru beserta teknik-teknik penyerangan baru yang digunakan oleh si penyusup.

3. Make a Crodit Attacker Logic
   Make a crodit Attacker Logic atau mengacaukan pola pikir penyusup merupakan suatu metode yang membuat pola pikir penyusup menjadi bingung dalam menghapi pola sistem jaringan komputer yang tidak sebenarnya.
4. Building System Defense
   Building System Defense atau Membangun Pertahanan merupakan suatu metode yang membuat suatu *Honeypot* yang dibangun akan memberikan pertahanan yang lebih bagus dikarenakan si penyusup tidak akan langsung melakukan penyerangan terhadap server sesungguhnya.
5. Know Your Enemy
   Know Your Enemy atau mengenal si penyusup merupakan suatu metode yang digunakan untuk mengetahui siapa si penyusup sesungguhnya, apa yang dikerjakan oleh si penyusup juga metode serta teknik yang digunakannya.
6. Safe The System
   Safe The System atau menyelamatkan sistem merupakan suatu metode yang digunakan untuk menjebak si penyusup sehingga penyusup berusaha tetap melakukan tindakan hanya pada sistem *Honeypot* sehingga server asli tetap dalam kondisi yang aman.
7. Hacking Process Prevention
   Hacking Process Prevention atau Mencegah Proses Hacking adalah suatu metode yang digunakan untuk membangun sebuah sistem pertahanan yang diharapkan akan mengurangi serangan terhadap proses *hacking*.

b. Klasifikasi Honeypot

Berdasarkan tingkat aktivitas yang dilakukan oleh seorang *intruder*, *Honeypot* dapat diklasifikasikan sebagai berikut:

1. Low-Interaction Honeypot
   Low-Interaction Honeypot atau *Honeypot* berinteraksi rendah adalah *Honeypot* yang dirancang untuk menciptakan *service* atau layanan palsu seperti pada komputer atau server yang sesungguhnya (Spitzner, 2003). Keuntungan dari *Honeypot* berinteraksi rendah adalah kesederhanaannya, karena dapat dengan mudah dibangun dan diperbaiki dengan resiko minimal. Kelemahan dari *Honeypot* berinteraksi rendah adalah *Honeypot* hanya mampu merekam informasi *log* yang terbatas dan hanya didesain untuk menangkap aktifitas yang sudah didefinisikan atau diketahui sebelumnya. Contoh *Honeypot* berinteraksi rendah adalah *qebek*, *sebek*, *honeyd*, *honeytrap*, *honeywall* (Honeynet, 2011).
2. High-Interaction Honeypot
   High-Interaction Honeypot atau *Honeypot* berinteraksi tinggi merupakan *Honeypot* yang benar-benar diimplementasikan dengan menggunakan sistem operasi dan aplikasi yang sesungguhnya tanpa adanya emulasi apapun seperti yang terdapat pada *Honeypot* berinteraksi rendah (Spitzner, 2003). Keuntungan dari *Honeypot* berinteraksi tinggi adalah mampu memberikan banyak informasi kepada pengguna terhadap aktifitas, perilaku dan tujuan dari *intruder* atau *attacker* tersebut. Kelemahan dari *Honeypot* berinteraksi tinggi adalah dapat meningkatkan resiko yang tinggi terhadap jaringan atau sistem yang sesungguhnya apabila *Honeypot* dapat diambil alih oleh *intruder* tersebut. Contoh dari *Honeypot* berinteraksi tinggi adalah *Honeynet* (Honeynet, 2011).

Daniel Of Services

Perkembangan teknologi internet dewasa ini bukan hanya menimbulkan dampak positif bagi kehidupan manusia, melainkan juga menimbulkan dampak negatif yang salah satunya adalah munculnya berbagai bentuk serangan yang mengancam keamanan komputer. Satu diantara bentuk serangan yang cukup popular didunia internet adalah *Denial of Service* (DoS). DoS merupakan salah satu serangan yang bertujuan untuk membuat komputer atau server yang menjadi target serangan menjadi *down* atau mati dikarenakan penyerang membanjiri komputer tersebut dengan pesan-pesan sampah dalam jumlah besar dan dilakukan secara terus menerus.

Serangan DoS juga mengalami perkembangan agar teknik serangan ini lebih sulit untuk dicegah. Jika sebelumnya DoS dilakukan hanya menggunakan satu komputer saja untuk menyerang, maka sekarang satu komputer penyerang akan menguasai beberapa komputer lain yang disebut dengan *zombie*. Kemudian komputer penyerang akan memerintahkan para *zombie* untuk melakukan serangan DoS ke satu target, sehingga efek yang dihasilkan bisa berkali-kali lipat dari

serangan DoS biasa. Teknik serangan ini dikenal dengan istilah *Distributed Daniel of Service* atau DDoS.

a. Metode Serangan Daniel of Service

DoS dan DDoS hanyalah sebutan untuk serangan pada jaringan atau komputer yang bertujuan untuk mematikan layanan dari suatu server atau komputer. Metode serangannya pun bervariasi, namun hampir tidak ada bedanya metode serangan pada DoS dan DDoS, hanya berbeda pada jumlah penyerangnya saja. Berikut ini adalah contoh metode serangan yang ada pada DoS dan DDoS:

1. SYN Flood

Pada protokol TCP dikenal *Three-Way Handshake*, yaitu sebelum terjadi koneksi antara kedua belah pihak, maka harus dilakukan pengiriman 3 macam paket untuk menyatakan bahwa koneksi telah terbangun, yaitu *SYN*, *SYN-ACK*, dan *ACK*. Pada *SYN Flood Attack*, penyerang akan mengirim sebuah paket *SYN* yang alamat IP-nya tidak ada atau tidak dapat diakses oleh korban. Sehingga paket *SYN* ini akan bertahan di dalam *stack* pada komputer korban sampai *ACK* diterima. Pada kenyataannya, *ACK* tidak akan pernah diterima, sehingga paket *SYN* terus memenuhi *stack* pada korban. Hal ini bisa berakibat paket *SYN* yang sah tidak dapat diterima oleh korban.

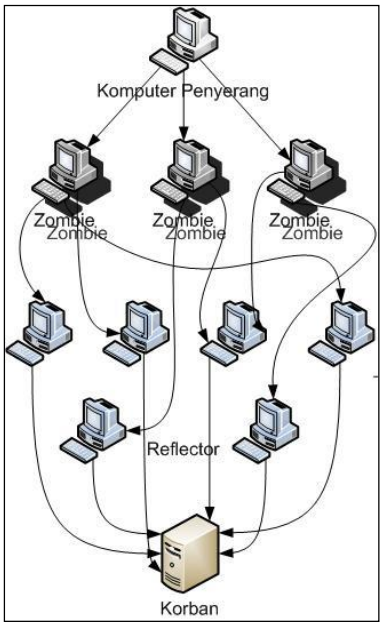
2. ICMP Flood

ICMP flood ini akan memanfaatkan alamat *broadcast* untuk melakukan penyerangan. Penyerang akan mengirimkan paket *ICMP request* ke alamat *broadcast* dari suatu jaringan, sehingga semua komputer yang berada pada jaringan tersebut akan menerima paket *ICMP request* tersebut dan seharusnya akan mengirimkan paket *ICMP reply* ke pengirim. Tetapi sebelumnya alamat IP pengirim sudah diubah menjadi alamat IP komputer target. Sehingga komputer-komputer yang mendapat paket *ICMP request* tadi akan mengirimkan paket *ICMP reply* ke komputer korban. Hal ini berakibat fatal jika ada banyak komputer pada suatu jaringan.

3. DNS Amplification Attack

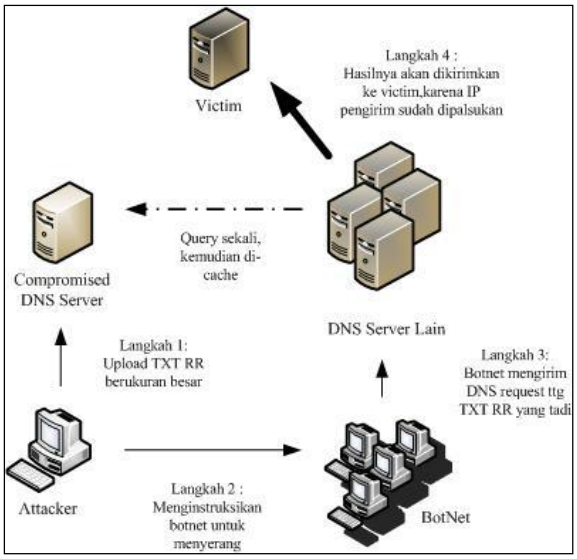
Serangan dengan tipe ini sebetulnya adalah pengembangan dari DDoS yang biasa disebut DRDoS atau *Distributed Reflector* DoS. Disebut reflektor karena selain komputer *zombie*, ada juga beberapa komputer lain yang menjadi perantara serangan seperti pada *ICMP Flood*, sehingga efeknya bisa berkali-kali lipat meskipun dengan jumlah komputer *zombie* yang sama, seperti terlihat pada gambar 1.

Pada DNS *Amplification Attack* biasanya penyerang akan menguasai dulu salah satu DNS Server yang ada dan DNS Server ini biasanya dijadikan rujukan bagi DNS Server yang lain. Kemudian penyerang akan membuat sebuah *Resource Record* (RR) baru yang memiliki ukuran sangat besar. Setelah itu penyerang akan memerintahkan komputer *zombie* untuk mengirimkan DNS *request* RR yang baru dibuat tadi.



Gambar 1. Distributed Reflector DoS (Peng, 2007)

Komputer *zombie* akan melaksanakan perintah tersebut dan meminta RR ke masing-masing DNS Server yang digunakan. Karena di DNS Server yang dituju *zombie* tidak terdapat RR yang diinginkan, maka DNS Server akan meminta RR tersebut ke DNS Server yang telah dikuasai. Sebelumnya ketika *zombie* mengirimkan DNS *request*, dia akan merubah alamat IP pengirimnya menjadi IP dari korban. Sehingga para DNS Server yang dimintai *request* akan mengirimkan jawabannya ke korban. Ilustrasinya bisa dilihat pada gambar 2.



Gambar 2. DNS Amplification Attack (Peng, 2007)

b. Metode Pertahanan Terhadap Denial of Service

Ada 4 macam kategori pertahanan terhadap DoS dan DDoS, yaitu *attack detection, attack prevention, attack source identification,* dan *attack reaction* (Peng, 2007). Pada *attack prevention*, penanganan serangan memfokuskan pada sebelum serangan terjadi dan bagaimana agar jangan sampai ada serangan yang dapat terjadi. Sedangkan pada *attack detection*, fokus utama adalah bagaimana agar suatu serangan dapat terdeteksi, untuk kemudian dilakukan penanganan lebih lanjut.

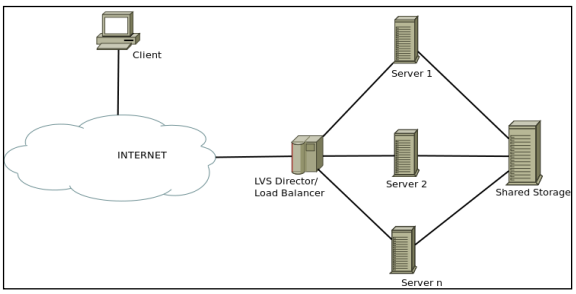
Fokus pada dua metode sebelumnya adalah pertahanan dari serangan. Pada *attack source identification*, yang menjadi fokus adalah bagaimana agar penyerangnya dapat ditemukan. Meskipun dari hasil yang ada masih sulit untuk diterapkan pada serangan yang termasuk dalam DDoS.

Jenis yang terakhir adalah *attack reaction.* Pada jenis ini lebih fokus kepada apa yang dilakukan setelah serangan terjadi dan bagaimana meminimalisir efeknya. Masing-masing metode memiliki kelebihan dan kekurangan. Tetapi pada paper (Peng, 2007) belum disebutkan penggunaan kombinasi dari jenis-jenis metode yang ada.

Linux Virtual Server

Linux Virtual Server (LVS) adalah salah satu cara untuk mengatur sejumlah server agar dapat memberikan layanan kepada pengguna secara bersama-sama. LVS mengarahkan koneksi dari klien ke server-server yang berbeda tergantung dari algoritma penjadwalan yang digunakan, dan membuat beberapa layanan yang berjalan paralel tampak sebagai sebuah layanan virtual yang berada pada sebuah alamat IP (Zhang, 2000). Selain itu, LVS juga mengatur penambahan ataupun pengurangan server. Jika ada salah satu server yang tiba-tiba tidak dapat memberikan layanan, maka LVS akan secara otomatis mengatur agar layanan yang ada pada server tersebut diberikan pada server yang lain.

Sebuah LVS terdiri dari satu atau lebih *Director/Load Balancer*, sejumlah klaster server yang bertugas untuk melayani pengguna, dan server yang digunakan sebagai tempat penyimpanan data bersama. Arsitektur LVS merupakan arsitektur *three-tier*. Untuk lebih jelasnya dapat dilihat pada gambar 3.



Gambar 3. Arsitektur Linux Virtual Server (Zhang, 2000)

Sebuah LVS memiliki satu alamat IP yang nantinya digunakan oleh klien untuk mengakses layanan yang diberikan. Ketika klien meminta suatu layanan, maka LVS *Director* akan mengarahkan permintaan tersebut ke salah satu server yang ada. Kemudian server yang ditunjuk akan merespon permintaan klien dengan opsi melewati LVS *Director* terlebih dahulu atau tidak.

a. Pengarahan Paket Data Linux Virtual Server

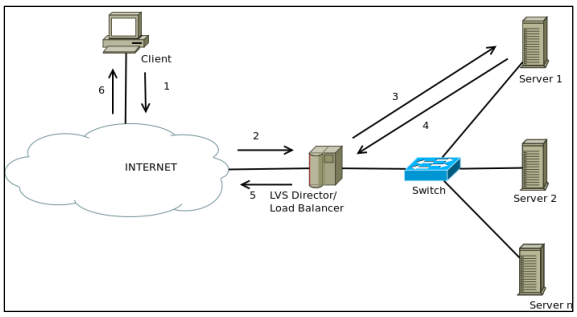
Linux Virtual Server memiliki beberapa metode yang digunakan untuk mengarahkan paket data yang dikirimkan oleh *client* ke *server*, yaitu *Network Address Translation*, *Tunneling*, dan *Direct Routing*.

1. Network Address Translation

Network Address Translation (NAT) atau *Linux Virtual Server - Network Address Translation* (LVS-NAT) merupakan metode yang digunakan untuk membuat sebuah IP lokal dapat mengakses internet dengan

memanfaatkan IP publik yang dimiliki dengan cara mengubah alamat IP pengirim menjadi alamat IP publik dari *router*. Sehingga, paket akan dikembalikan ke IP publik tersebut dan seterusnya dikirimkan kembali ke pengirim yang sesungguhnya.

Metode ini adalah metode yang pertama kali keluar dan paling mudah untuk diimplementasikan karena tidak membutuhkan konfigurasi khusus di sisi server. Selain itu *backend* server yang digunakan bisa terdiri dari berbagai macam sistem operasi apabila sistem operasi tersebut menggunakan protokol *TCP/IP stack*. Berikut merupakan ilustrasi *LVS* yang menggunakan metode *LVS-NAT*.



Gambar 4. Metode LVS – Network Address Translation (Zhang, 2000)

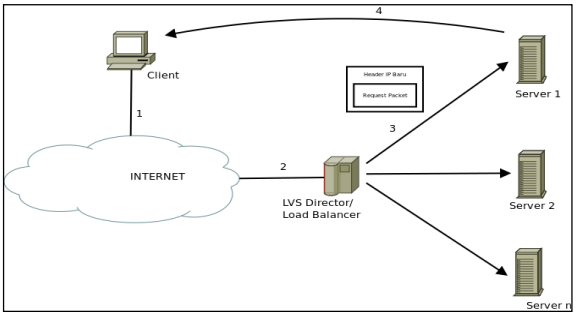
Ketika klien ingin mengakses layanan tertentu pada *LVS*, maka dia akan mengirimkan paket data dari *LVS Director/Load Balancer* (1, 2) melalui internet. Kemudian *LVS Director/Load Balancer* akan mengarahkan paket tersebut ke salah satu server fisik yang ada dengan mengubah IP tujuan, *port* tujuan dan mencatat koneksinya ke dalam *hash table* (3). Server akan merespon permintaan klien dan mengirimkannya melalui *LVS Director/Load Balancer* (4). *LVS Director/Load Balancer* memeriksa *hash table* koneksi miliknya untuk menentukan akan diarahkan kemana paket data yang diterima dari server. Kemudian paket tersebut akan ditulis ulang alamat IP asalnya sehingga tampak bahwa *LVS Director/Load Balancer* mana yang mengirim (5, 6).

Kelebihan dari metode ini adalah kemudahannya untuk diimplementasikan, sehingga banyak pengguna LVS pemula yang menggunakan metode ini untuk lebih memahami cara kerja LVS. Akan tetapi untuk LVS yang digunakan dalam proses produksi, metode ini jarang dipakai. Hal itu dikarenakan efek penulisan ulang alamat IP tujuan dan asal membutuhkan waktu yang lebih lama dibandingkan metode-metode lainnya, apalagi jika jumlah klien yang mengakses dan *backend* server semakin banyak.

2. Tunneling

Tunneling atau *Linux Virtual Server – Tunneling* (LVS-TUN) merupakan suatu teknik yang digunakan untuk membungkus sebuah paket IP ke dalam paket IP. Sehingga suatu paket IP yang ditujukan untuk suatu alamat bisa dibungkus kemudian ditambahkan informasi IP Tujuan yang baru agar paket tersebut dapat diarahkan ke tempat lain. Berbeda dengan *LVS-NAT*, pada *LVS-TUN* tidak terjadi perubahan *header* paket IP, tetapi menambahkan *header* ke dalam sebuah paket IP yang mengakibatkan ukuran paket semakin besar.

Alur paket pada *LVS-TUN* dapat dilihat pada gambar 2.5. Ketika klien meminta suatu layanan tertentu, dia akan mengirimkan paket ke *LVS Director/Load Balancer*. Kemudian, *LVS Director/Load Balancer* akan membungkus paket IP tersebut dengan *header* IP yang berisi informasi alamat IP *backend* server tujuan dan mengirimkannya ke server tujuan.



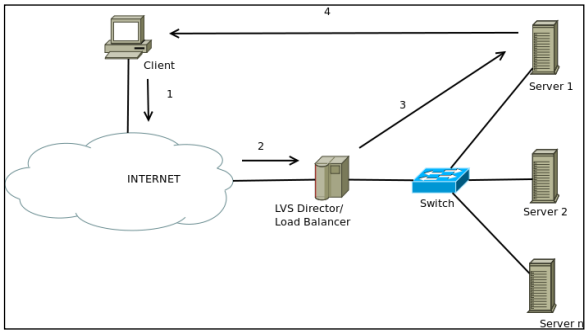
Gambar 5. Metode LVS – Tunneling (Zhang, 2000)

Tetapi ketika *backend* server ingin mengirimkan paket respon dari permintaan klien, *backend* server tidak mengirimkannya melalui *LVS Director/Load Balancer*. *Backend* server akan langsung mengirimkan ke klien.

Keuntungan dari penggunaan LVS-TUN ini adalah *backend* server tidak harus terletak pada jaringan yang sama dengan *LVS Director/Load Balancer*. Bahkan, *backend* server dapat diletakkan di tempat yang berbeda secara geografis, selama masih terhubung dengan internet. Namun, LVS-TUN ini hanya dapat digunakan oleh sistem operasi yang mendukung *IP Tunneling* seperti Linux.

3. Direct Routing

Metode lain yang dapat digunakan selain LVS-NAT dan LVS-TUN adalah *Direct Routing* atau *Linux Virtual Server - Direct Routing* (LVS-DR). Cara kerja dari LVS-DR ini hampir sama dengan LVS-TUN, hanya saja pada LVS-DR tidak dilakukan proses enkapsulasi paket IP. LVS-DR memanfaatkan ARP untuk meneruskan paket dari *LVS Director/Load Balancer* ke *backend* server. Apabila LVS-DR digunakan, maka *LVS Director/Load Balancer* dan *backend* server akan berbagi sebuah alamat IP virtual yang sama.



Gambar 6. Metode LVS – Direct Routing (Zhang, 2000)

Ketika ada paket datang di *LVS Director/Load Balancer* dan sesuai dengan layanan yang disediakan oleh LVS, *LVS Director/Load Balancer* akan mengubah MAC *Address* yang tertera di dalam paket dan mentransmisikan ulang ke LAN yang terhubung dengannya sehingga paket akan sampai di *backend* server yang memiliki MAC *Address* yang tertulis.

Karena LVS-DR ini memanfaatkan MAC *Address* untuk mengirimkan paket ke *backend* server, maka antara *backend* server yang ada dengan *LVS Director/Load Balancer* harus terhubung ke dalam 1 segmen LAN yang sama, baik itu menggunakan *switch* atau *hub*. Kemudian sama seperti LVS-TUN, *backend* server akan langsung mengirimkan paket responnya ke klien melalui *gateway* yang digunakan. Ilustrasi dari alur kerja LVS-DR dapat dilihat pada gambar 6.

b. Pembagian Beban Linux Virtual Server

Keuntungan lain dari penggunaan LVS adalah kemampuannya untuk membagi beban ke sejumlah *backend* server yang ada, sehingga kinerjanya menjadi lebih baik daripada penggunaan virtual server yang lain. LVS memiliki empat algortima pembagian beban yang dapat digunakan, yaitu *round robin, weighted round robin, least connection,* dan *weighted least-connection.*

1. Round Robin

Round Robin adalah metode pembagian beban koneksi yang paling sederhana pada LVS. Dengan menggunakan *round robin* ini, beban server ataupun bobot server tidak diperhitungkan. Yang dilakukan oleh *LVS Director/Load Balancer* ketika ada koneksi baru yang datang adalah memberikannya ke server sesuai dengan urutannya. Yang dilakukan oleh metode *round robin* pada LVS sebetulnya bisa dikatakan sama dengan yang dilakukan oleh DNS Server jika ada *record* sama yang memiliki IP yang berbeda.

2. Weighted Round Robin

Pada algoritma *round robin* tidak diperhitungkan mengenai bobot dari masing-masing server. Setiap server dianggap dapat menanggung beban kerja yang sama. Hal ini tidak bermasalah jika spesifikasi server-server tersebut sama. Tetapi jika spesifikasi server-server tersebut beraneka ragam, maka server yang seharusnya hanya dapat menangani klien lebih sedikit mendapatkan beban yang sama dengan klien yang dapat melayani klien yang lebih banyak.

Untuk itu ada metode lain yang disebut dengan *Weighted Round-Robin*. Dengan metode ini, setiap server memiliki *weight*/bobot yang berbeda-beda. Server yang lebih kuat dapat diberi bobot yang lebih besar.

Secara garis besar, algoritma ini hampir sama dengan *round-robin* biasa. Hanya saja pemilihan server yang dilakukan tidak hanya secara berurutan, tetapi juga mempertimbangkan server dengan bobot terbesar saat itu. Setiap server yang telah memberikan koneksi baru dikurangi bobotnya. Apabila bobot server telah mencapai 0, maka bobotnya akan kembali ke nilai awal.

3. Least Connection

Least-connection membagi koneksi berdasarkan jumlah koneksi yang aktif pada saat itu. *LVS Director/Load Balancer* akan memberikan koneksi baru pada server yang memiliki jumlah koneksi aktif paling sedikit. Hal ini dilakukan agar koneksi yang memiliki waktu lama dapat terdistribusi merata ke server-server yang ada.

4. Weighted Least Connection

Sekilas, *Least-connection* tampak sempurna. Tetapi *Least-connection* akan memiliki kinerja yang tidak maksimal jika *backend* server yang ada memiliki spesifikasi yang berbeda karena pembagian beban antar server tidak merata. Server dengan spesifikasi yang lebih canggih dan dapat menangani banyak koneksi justru malah mendapatkan sedikit koneksi baru karena cepatnya dia menangani *request* klien.

Karena itulah ada metode lain yang memberikan bobot untuk masing-masing server. Jadi, server dengan spesifikasi yang lebih canggih akan mendapatkan bobot yang lebih besar. Dengan bobot yang lebih besar, maka server tersebut juga mendapatkan koneksi yang lebih banyak, karena memang server tersebut mampu.

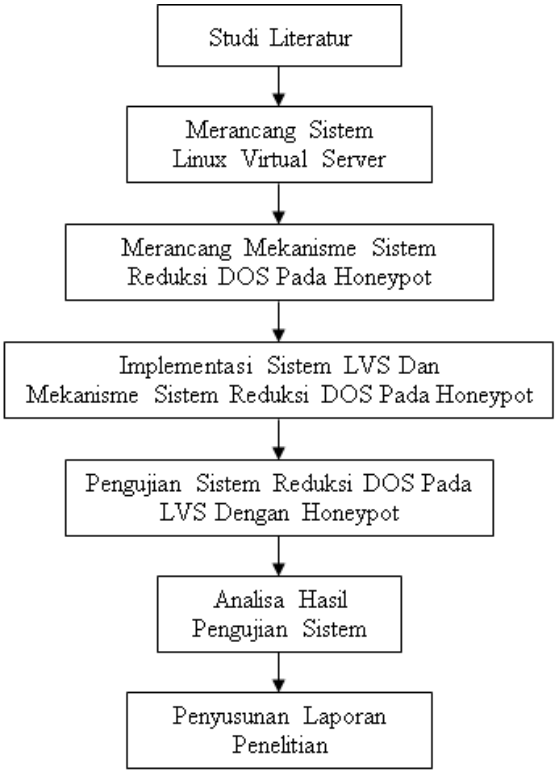
c. Modul Linux Virtual Server

Secara fungsionalitasnya, *Linux Virtual Server* memiliki dua bagian penting yang mempunyai tugas utama masing-masing, yaitu *IP Virtual Server* dan *IP Virtual Server Administration*. IP Virtual Server (ip_vs) merupakan modul *kernel* dari *Linux Virtual Server* yang bertugas untuk melakukan pengaturan pengarahan *request*, penjadwalan, dan penyimpanan data. Sedangkan *IP Virtual Server Administration* (*ipvs_administration*) merupakan bagian *kernel* modul kedua *Linux Virtual Server* yang bertugas untuk pengaturan manajemen *Linux Virtual Server* yang tersedia.

I. Metodologi

Kegiatan Penelitian

Pada sub-bab kegiatan penelitian ini dijelaskan tentang bagaimana kegiatan sebelum, sedang dan setelah penelitian ini dilakukan. Kegiatan yang dilakukan meliputi pencarian Studi Literatur, merancang sistem LVS, merancang mekanisme sistem reduksi DoS, implementasi sistem, pengujian sistem, analisa hasil pengujian, dan penulisan laporan penelitian. Berikut gambaran dan penjelasan tentang rancangan penelitian yang dilakukan selama melaksanakan penelitian ini, yaitu:



Gambar 7 Kegiatan Penelitian

1. Studi Literatur

Kegiatan yang dilakukan pada Studi Literatur ini adalah mencari gambaran tentang konsep penelitian ini berdasarkan dari beberapa referensi ilmiah yang meliputi buku, paper/jurnal, Tesis, Prosiding seminar nasional, Tugas Akhir/Skripsi, hingga menemukan suatu kesimpulan yang menjadi dasar pada penelitian ini dan hasil pencarian Studi Literatur secara lengkap tersaji pada bab Tinjauan Pustaka yang akan dijadikan dasar utama dalam melakukan penelitian ini.

2. Merancang Sistem Linux Virtual Server

Pada tahap ini penulis melakukan suatu analisis berdasarkan dari kesimpulan-kesimpulan yang ditemukan pada Studi Literatur, dan selanjutnya dari hasil analisis tersebut dibuatlah suatu rancangan sistem Linux Virtual Server yang akan digunakan sebagai skenario pengujian reduksi DoS pada penelitian ini. Selain itu, dibuat juga skenario pengujian yang didalamnya berisi langkah-langkah beserta jenis serangan yang dilakukan untuk menguji keberhasilan dari rancangan kolaborasi sistem yang dibuat, sehingga tujuan dari penelitian ini bisa tercapai dengan baik.

3. Merancang Mekanisme Sistem Reduksi DoS Pada Honeypot

Pada tahap ini penulis melakukan suatu analisis berdasarkan dari kesimpulan-kesimpulan yang ditemukan pada studi literatur, yaitu merancang mekanisme bagaimana mereduksi serangan DoS dan DDoS dengan cara mengalihkan serangan tersebut ke sebuah komputer atau server palsu yang disebut dengan *Honeypot*. Mekanisme yang dimaksud berupa rancangan arsitektur sistem pengalihan serangan dan algoritma reduksi serangan DoS maupun DDoS dengan memanfaatkan Honeypot sebagai komputer atau server palsu tujuan pengalihan serangan.

4. Implementasi Sistem LVS

Pada tahap ini dilakukan implementasi sesuai dengan rancangan sistem yang telah dibuat. Implementasi dilakukan pada lingkungan virtualisasi dengan menggunakan software *VMware* sebagai lingkungan virtualisasinya, yang didalamnya terinstal sistem operasi Linux Ubuntu dan sudah diinstal software-software pendukung sesuai dengan kebutuhan sistem yang digunakan dalam melakukan penelitian.

5. Pengujian Sistem Reduksi DoS

Pada tahap pengujian, pengujian dilakukan sesuai dengan rancangan sistem yang telah dibuat dengan menambahkan skenario pengujian didalamnya. Tujuan dilakukan pengujian pada tahap ini adalah untuk membuktikan apakah kontribusi penelitian yang sudah ditentukan diawal penelitian telah tercapai dengan baik sesuai dengan teori penunjang yang terdapat pada Studi Literatur, rancangan dan skenario pengujian yang dibuat serta metode maupun teknik implementasi yang digunakan.

6. Penulisan Laporan

Kegiatan yang dilakukan pada tahap ini adalah mendokumentasikan hasil penelitian ini kedalam suatu bentuk laporan ilmiah sesuai dengan aturan dan kaidah penyusunan penulisan ilmah yang terdapat pada buku panduan penyusunan penelitian. Dokumentasi yang dilakukan meliputi penulisan konsep dan tujuan penelitian, studi literatur, analisa dan desain, implementasi, uji coba, kesimpulan dan saran, serta daftar pustaka yang kemudian dibukukan menjadi sebuah buku penelitian yang terpublikasi.

Langkah-Langkah Penelitian

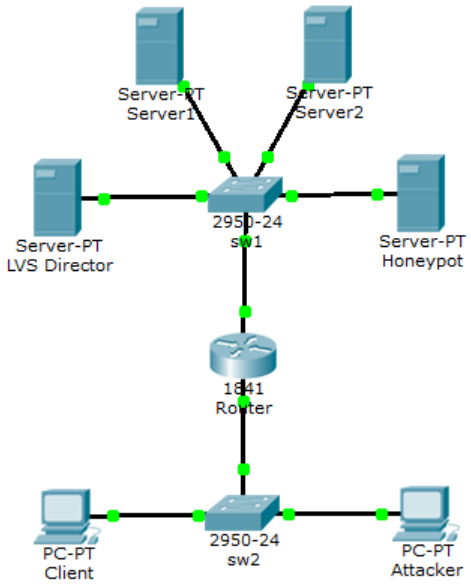
Sub bab langkah-langkah penelitian merupakan penerapan yang dilakukan setelah melalui tahapan rancangan kegiatan penelitian. Pada sub bab ini akan dijelaskan desain sistem dan skenario pengujian yang dibuat pada penelitian ini.

Arsitektur Sistem

Lingkungan pengujian pada penelitian ini dilakukan dalam bentuk virtualisasi menggunakan software *VMwar workstation* dengan arsitektur sistem yang terdiri dari komputer client, *attacker*, *router*, *LVS director*, *Honeypot*, dan dua buah server web. Penjelasan tugas dan spesifikasi dari masing-masing komputer adalah sebagai berikut:

1. Komputer client disini tidak memiliki spesifikasi khusus dikarenakan komputer ini hanya bertugas untuk melakukan pengujian pengiriman paket yang tidak berisi usaha penyerangan terhadap server.
2. Komputer *attacker* adalah komputer yang didalamnya terdapat software-software *exploit* yang ter-*include* pada sistem operasi Linux Backtrack, bertugas untuk melakukan usaha-usaha penyerangan berupa serangan DoS dan DDoS terhadap komputer server web atau target.
3. Komputer *router* adalah komputer yang bekerja menggunakan sistem operasi linux ubuntu yang bertugas untuk meneruskan paket dari client dan *attacker* menuju komputer server baik yang berisi paket data yang sah maupun paket data yang berisi serangan DoS ataupun DDoS.
4. Komputer *LVS Director* adalah komputer yang bertugas untuk mengarahkan paket data yang berisi serangan DoS ataupun DDoS ke komputer *Honeypot*. Komputer *LVS Director* bekerja pada sistem operasi linux yang membutuhkan konfigurasi khusus didalamnya. Penjelasan detail konfigurasi pada komputer *LVS Director* dijelaskan pada bagian lampiran penelitian ini.
5. Komputer *Honeypot* bertindak sebagai server palsu yang bertugas untuk memberi respon palsu terhadap serangan DoS atau DDoS yang dikirimkan oleh komputer client yang sudah terdeteksi dan diarahkan oleh komputer *LVS director* menuju komputer *Honeypot*.
6. Komputer server web adalah komputer yang bertugas menyediakan layanan web dan merupakan komputer yang digunakan

sebagai target penyerangan DoS atau DDoS yang dilakukan oleh komputer *attacker*.



Gambar 8 Arsitektur Sistem

Skenario Pengujian

Skenario pengujian dibuat dengan tujuan sebagai gambaran langkah-langkah yang dilakukan dalam pembuktian dan evaluasi terhadap kinerja sistem yang dibuat. Sehingga diperlukan parameter-parameter penilaian yang akan digunakan untuk mengukur efektifitas dari metode-metode yang digunakan. Adapun paramater-parameter penilaian yang digunakan dalam pengujian pada penelitian ini antara lain:

1. Repy Rate

Reply Rate adalah jumlah respon dari server yang diterima oleh klien dalam satu waktu periode pengujian.

2. Waktu Respon Rata-rata

Waktu respon rata-rata merupakan waktu yang dibutuhkan klien untuk mengirim hingga menerima respon dari server. Parameter ini menunjukkan lamanya pesan respon diterima oleh klien sebelum dan sesudah ada sistem pengarahan paket ketika terjadi serangan.

3. Request Yang Gagal

Request yang gagal meliputi presentase *request* yang gagal dikirim maupun *request* yang terkirim tetapi respon tidak diterima klien dibandingkan dengan keseluruhan paket yang dikirim. Parameter ini juga penting untuk menunjukkan kinerja dari metode yang akan diimplementasikan.

4. Pembagian Beban Kerja Penggunaan CPU

Parameter ini digunakan untuk mengukur kondisi *Load Balancer* sebelum dan sesudah ada sistem pengarahan paket, apakah ada lonjakan beban yang besar setelah penerapan pengarahan paket atau tidak.

II. Hasil dan Pembahasan

Implementasi

Pada tahap implementasi ini, terdapat beberapa tahapan yang dilakukan oleh penulis, antara lain menentukan lingkungan pengembangan yang terdiri dari perangkat keras dan perangkat lunak yang digunakan, beberapa konfigurasi perangkat lunak dan jaringan yang digunakan, serta implementasi masing-masing komponen yang digunakan sesuai dengan arsitektur sistem yang telah dibuat.

Lingkungan Pengembangan

Lingkungan pengembangan sistem pada penelitian ini dilakukan pada lingkungan virtual dengan menggunakan perangkat lunak *VMware Workstation* sebagai tools virtualisasinya. Pada penelitian ini, pengujian dilakukan dengan menggunakan enam komputer virtual baik sebagai server, *LVS Director*, *Honeypot*, *Router*, *Attacker*, maupun *Client* serta tiga *virtual switch* yang masing-masing spesifikasinya sebagai berikut:

1. Server 1 dan 2
   a) Memori 1 GB
   b) HD 20 GB
   c) SO Linux Ubuntu 14.04 LTS
2. LVS Director
   a) Memori 1 MB
   b) HD 10 GB
   c) SO Linux Ubuntu 14.04 LTS
3. Honeypot
   a) Memori 512 MB
   b) HD 10 GB
   c) SO Linux Ubuntu 12.04 LTS
4. Router dan Internet
   a) Memori 64 MB
   b) HD 10 GB
   c) SO Linux Ubuntu 12.04 LTS
   d) iptables
5. Attacker dan Client
   a) Memori 256 MB
   b) HD 20 GB
   c) SO Linux Backtrack R5

Selain menggunakan hardware seperti yang sudah disebutkan sebelumnya, pada penelitian ini juga menggunakan beberapa software yang digunakan untuk proses pengujian sistem yang dibangun. Tujuannya untuk mengetahui tingkat keberhasilan secara fungsionalitas dan performa sistem jika menggunakan skenario pengujian yang diusulkan. Berikut beberapa software yang digunakan beserta fungsinya:

1. Autobench 2.1 dan Httperf 9.0

Software *Autobench* dikolaborasikan dengan *Httperf* dengan tujuan agar *Httperf* dapat berjalan secara otomatis berulang kali untuk mengukur performa Server yang digunakan. Parameter pengujian performa yang dinilai adalah *reply rate, packet loss rate, response time* baik *inbound* maupun *outbound.*

2. Apache Benchmark 2.3

Apache Benchmark digunakan untuk mengukur performa server dengan parameter pengujian performa yang dinilai adalah *reply rate, packet loss rate, response time* baik *inbound* maupun *outbound*. Tujuannya adalah untuk membandingkan dan memastikan hasil pengujian performa yang dihasilkan oleh kolaborasi software *Autobench* dan *Httperf*.

3. SAR

SAR digunakan untuk mengukur kinerja Server selama server bekerja pada rentang waktu tertentu, tujuannya untuk mengetahui penggunaan CPU dan Memory komputer Server.

Implementasi Sistem Deteksi Intrusi

Langkah pertama yang dilakukan pada tahap implementasi ini selain menyiapkan lingkungan pengembangan adalah dengan mengimplementasikan sistem deteksi intrusi. Tujuannya adalah supaya dapat mendeteksi apakah paket yang dikirimkan merupakan *request* yang benar ataukah *request* yang mempunyai potensi sebagai intrusi atau serangan. Seperti konsep yang diusulkan, serangan yang dimaksud pada penelitian ini adalah serangan *Daniel of Service* menggunakan metode *SYN Flooding*, artinya jika terdapat *request* yang dikirimkan dan terdeteksi sebagai serangan *Daniel of Service* maka sistem akan mengalihkan paket tersebut ke server palsu atau *Honeypot* agar dampak dari serangan tersebut pada real server dapat tereduksi. Oleh karena itu perlu dilakukan pemilihan *rule options* pada sistem deteksi serangan yang tepat agar mampu mendeteksi serangan *Daniel of Service SYN Flooding* ini dengan baik.

Sistem Deteksi Intrusi sudah menyediakan banyak *rule options* yang dapat digunakan untuk mendeteksi serangan *Daniel of Service* termasuk yang menggunakan metode *SYN Flood*, yaitu *rule options detection_filter*. Rule *detection_filter* adalah *rule options* Sistem Deteksi Intrusi yang digunakan untuk mendeteksi serangan *SYN Flood* dengan baik.

Namun setelah dilakukan pengujian yang menggunakan serangan *SYN Flood* dengan jumlah yang sangat banyak, Rule *detection_filter* ternyata memiliki kelemahan yaitu tidak dapat membedakan *request* yang sah atau yang bukan berpotensi serangan *SYN Flood*. Oleh karena itu perlu ditambahkan *event processing rate_filter* kedalam rule options *detection_filter* agar hasil deteksi yang dilakukan lebih akurat. Hal ini disebabkan, *request* yang terjadi beberapa kali selama beberapa detik akan dideteksi sebagai serangan. Apabila setelah terdeteksi sebagai serangan dan kemudian *rate*-nya turun di bawah batas maksimum yang sudah ditentukan, maka berikutnya *request* tersebut tidak akan lagi terdeteksi sebagai serangan.

Pada gambar 9 berikut adalah *rule options* yang digunakan pada Sistem Detekti Intrusi penelitian ini.

```
pass tcp $EXTERNAL_NET any ->
$HTTP_SERVERS 80
(
        msg:"DDOS synflood"; \
        flow:to_server; \
        flags:S; \
        sid:1000001; rev: 1
)
rate_filter \
        gen_id 1, sig_id 1000001, \
        track by_src, count 4000, seconds
1, \
        new_action alert, timeout 0
```

Gambar 9 Rule Options Sistem Deteksi Intrusi

Penjelasan dari *rule options* pada gambar 9 diatas adalah Sistem Deteksi Intrusi tetap akan mengarahkan *request* yang berisi paket *SYN* menuju server apabila hanya terdapat satu *request* yang masuk dalam satu detik. Namun apabila dalam waktu satu detik terdapat *request* menuju server dengan jumlah lebih dari sama dengan 4.000 *request*, maka Sistem Deteksi Intrusi akan mengeluarkan *alert* yang berarti

request yang masuk berpotensi serangan *Denial of Service* atau *SYN Flooding*.

Selanjutnya untuk mengatasi agar *alert* yang dihasilkan tidak membebani server akibat banyaknya *alert* dengan tipe yang sama dikeluarkan, maka penambahan *event_filter* pada *rule options* tersebut bertujuan untuk membatasi *alert* yang dihasilkan terhadap semua *request* yang berpotensi serangan dengan jeda waktu selama 30 detik. Sehingga apabila terdapat *alert* yang sama dalam waktu kurang dari 30 detik setelah *alert* tersebut dikeluarkan, maka selanjutnya Sistem Deteksi Intrusi tidak akan mengeluarkan *alert* untuk *request* tersebut.

Sistem Deteksi Intrusi akan menganggap *alert* tersebut sama apabila *sig_id* atau *signature id* yang terdapat pada *alert* bernilai sama. Setiap *alert* yang dihasilkan oleh Sistem Deteksi Intrusi akan memiliki *sig_id* yang berbeda kecuali *alert* tersebut dihasilkan berdasarkan *request* yang berasal dari *ip address* asal yang sama, tujuan *ip address* dan *port* yang sama, jenis *request* yang sama, dan dikirimkan berulang kali dengan rentang waktu yang berdekatan. Oleh karena itu, jika terdapat *alert* yang memiliki *sig_id* yang sama maka Sistem Deteksi Intrusi hanya akan mengeluarkan *alert* sebanyak satu kali dengan tujuan supaya tidak terjadi *alert flooding* pada ruang penyimpanan Sistem Deteksi Intrusi.

Implementasi Honeypot

Honeypot merupakan suatu sistem yang dibuat menyerupai sistem aslinya yang bertugas untuk mendeteksi dan menganalisis *request* yang masuk (Spitzner, 2003). Artinya, jika terdapat *request* yang masuk berasal dari alamat IP yang terdeteksi sebagai penyerang maka *request* tidak akan diteruskan ke *real server* melainkan dialihkan ke *Honeypot* dengan tujuan agar *real server* terhindar dari *request* yang berpotensi serangan tersebut.

Konsep yang diusulkan pada penelitian ini tentang bagaimana mengalihkan serangan ke *Honeypot* adalah dengan melakukan kolaborasi dengan Sistem Deteksi Intrusi yang bertugas untuk mendeteksi alamat IP dari *request* yang berpotensi serangan. Seperti yang dijelaskan pada bagian Implementasi Sistem Deteksi Intrusi diatas bahwa selain bertugas melakukan deteksi dan memastikan kalau *request* tersebut berpotensi serangan, Sistem Deteksi Intrusi juga bertugas untuk mencatat alamat IP dari *request* yang berpotensi serangan tersebut yang nantinya akan dijadikan referensi *Linux Virtual Server* dalam memutuskan apakah *request* tersebut diteruskan ke Komputer Server yang sesungguhnya atau dialihkan ke *Honeypot* untuk dianalisis tujuan dari *request* tersebut dikirimkan.

Yang perlu dilakukan agar *Linux Virtual Server* dapat melakukan pengalihan atau penerusan *request*, maka perlu dilakukan perubahan aturan yang terdapat pada modul kernel *Linux Virtual Server Scheduller*. Karena metode penjadwalan proses *Linux Virtual Server* yang digunakan pada penelitian ini adalah *Round Robin*, maka modul kernel yang digunakan dan dirubah adalah modul kernel *ip_vs_rr*. Pada gambar 10 dapat dilihat perubaha aturan yang dilakukan pada modul kernel *ip_vs_rr*.

```
int is_blacklist;

ip_vs_fill_iphdr(svc -> af,
  skb_network_header(skb), &iph);

is_blacklist =
  ip_vs_honey_is_blacklist(iph.saddr.ip);

dest = list_entry(q,
  struct ip_vs_dest, n_list);

if((is_blacklist && !dest -> honeypot)
  || (!is_blacklist && dest -> honeypot))
{
     q = q -> next;
     continue;
}
```

Gambar 10 Aturan Pengalihan Request Pada Modul LVS Scheduller

Tujuan dari aturan itu dibuat adalah jika terdapat *request* yang masuk berasal dari alamat IP yang terdapat pada daftar *blacklist*, maka *request* tersebut akan dialihkan ke *Honeypot*, begitu juga sebaliknya apabila *request* yang masuk berasal dari alamat IP yang tidak terdapat pada daftar *blacklist*, maka request diteruskan ke *real server* atau komputer server yang sesungguhnya.

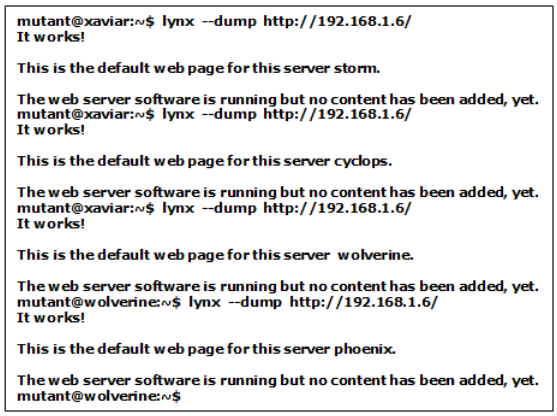
Pembahasan

Hasil yang sudah dicapai pada penelitian ini dapat ditunjukkan pada hasil pengujian yang dilakukan, yaitu pengujian secara fungsionalitas berdasarkan arsitektur yang diusulkan. Pengujian ini dilakukan untuk membuktikan apakah fungsi-fungsi dasar pada sistem ini berjalan sesuai dengan konsep yang diusulkan,

yaitu sistem yang dibangun mampu mengabaikan *request* yang dikirimkan ke server tidak berpotensi serangan *DOS* maupun *DDOS*, dan juga mampu mendeteksi *request* yang dikirimkan berpotensi serangan *DOS* maupun *DDOS* kemudian mengarahkan *request* tersebut ke *Honeypot* dengan tujuan untuk mereduksi dampak yang ditimbulkan oleh serangan *DOS* maupun *DDOS* tersebut. Pengujian yang dilakukan meliputi pengujian pada protokol HTTP menggunakan *web browser* dari komputer *Client* dan komputer *Attacker*, dan pengujian serangan *DOS* maupun *DDOS*.

Pengujian Pada Komputer Client

Pengujian pada komputer client dilakukan dengan cara mengakses halaman web pada komputer server dengan menggunakan *web browser lynx* sebanyak empat kali. Berikut ini adalah gambar dari proses pengujian pada komputer client.

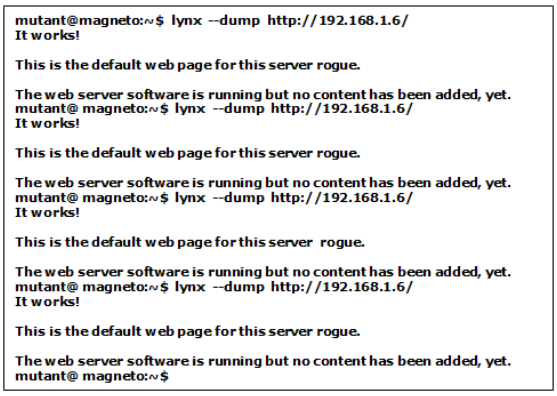


Gambar 11 Hasil Pengujian Komputer Client

Pada hasil pengujian komputer *Client* yang terdapat pada gambar 11 diatas, terlihat jelas bahwa *request* yang dikirimkan sebanyak empat kali telah mendapatkan *reply* dari komputer yang berbeda, namun tidak tampak mendapatkan *reply* dari komputer *Honeypot* sama sekali. Hal ini menunjukkan bahwa *LVS Director* yang dibangun telah bekerja dengan baik secara fungsionalitas.

Pengujian Pada Komputer Attacker

Pengujian pada komputer *Attacker* dilakukan dengan cara yang sama dengan yang dilakukan oleh komputer Client, yaitu mengakses halaman web pada komputer server dengan menggunakan *web browser lynx* sebanyak empat kali. Tujuan dari pengujian ini adalah untuk membuktikan apakah hasil pengujian yang dilakukan oleh komputer yang berasal dari alamat IP ter-*blacklist*, *request*-nya akan dialihkan ke komputer *Honeypot* sepenuhnya. Berikut ini adalah gambar dari proses pengujian pada komputer *Attacker*.



Gambar 12 Hasil Pengujian Komputer Attacker

Pada hasil pengujian komputer *Attacker* yang terdapat pada gambar 12 diatas, terlihat jelas bahwa *request* yang dikirimkan sebanyak empat kali telah mendapatkan *reply* dari komputer yang sama yaitu komputer *Honeypot*, dan tidak satupun *request* yang dikirimkan mendapatkan *reply* dari komputer Server yang sebenarnya. Hal ini menunjukkan bahwa *LVS Director* yang dibangun telah bekerja dengan baik secara fungsionalitas, yaitu mampu mengalihkan semua *request* yang masuk dari komputer dengan alamat IP ter-*blacklist*.

Pengujian Sistem Terhadap Serangan Denial of Service Attack Terdistribusi

Pengujian sistem terhadap serangan *Denial of Service Attack* terdistribusi dilakukan dengan tujuan untuk memastikan apakah sistem pengalihan *request* yang sudah dibangun dapat berfungsi dengan baik jika terdapat sejumlah *request* yang datang secara bersamaan dengan jenis *request* yang sama, waktu yang berdekatan, dan tujuan *request* yang sama. Pengujian pengalihan *request* terhadap alamat IP yang ter-*blacklist* sudah berhasil dilakukan dengan baik seperti yang sudah dijelaskan pada subbab 5.1.2 sebelumnya. Oleh karena itu dibutuhkan pengujian pengalihan *request* berikutnya menggunakan metode pengiriman *request* yang berbeda yang merupakan inti dari penelitian ini, yaitu *Distributed Daniel of Service Attack*.

Skenario pengujian yang dilakukan adalah pengiriman *request* yang dilakukan secara bersamaan oleh lima komputer client atau penyerang dalam hal ini kepada Server target. Yang dilakukan oleh kelima penyerang tersebut adalah mengirimkan *request* berupa paket *SYN* dengan jumlah yang sangat besar ditujukan ke port 80 server target secara bersamaan. Berikut perintah yang dilakukan oleh masing-masing komputer penyerang kepada server target.

```
root@magneto:~#  hping3  -i  u100  -a
192.168.1.250 -S -q -p 80
root@mystique:~#  hping3  -i  u100  -a
192.168.1.251 -S -q -p 80
root@sabretooth:~#  hping3  -i  u100  -a
192.168.1.252 -S -q -p 80
root@toad:~#  hping3  -i  u100  -a
192.168.1.253 -S -q -p 80
root@pyro:~#  hping3  -i  u100  -a
192.168.1.254 -S -q -p 80
```

Perintah tersebut merupakan perintah untuk mengirimkan paket *SYN* sebanyak 1.000 paket per detik ditujukan ke *port* 80 server target yang berasal dari alamat IP masing-masing komputer penyerang. Selanjutnya, hasil yang ditunjukkan dari pengiriman *request* yang dilakukan oleh kelima komputer penyerang terlihat pada gambar 13 dibawah ini.

```
Every 2.0s: ipvsadm

IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn Honeypot
ipvs_get_dests
TCP  192.168.1.6:www rr
  -> 192.168.1.3:www              Route   1      0          40473      0
  -> 192.168.1.4:www              Route   1      0          46464      0
  -> 192.168.1.5:www              Route   1      0          158052     1
```

Gambar 13 Hasil Serangan DDOS Pada LVS Director

Gambar 5.3 merupakan hasil kerja dari modul kernel *ipvs_administration* yang terdapat pada *LVS Director*. Dari gambar tersebut tampak dengan jelas bahwa jumlah koneksi terbesar terdapat pada alamat IP 192.168.1.5 yang merupakan alamat IP komputer *Honeypot*. Jumlah koneksi yang terdapat pada komputer server taget hanya kurang lebih 33% dari jumlah koneksi yang terdapat pada komputer *Honeypot*. Artinya, sistem yang dibangun mampu mengatasi serangan *Daniel of Service Attact* terdistribusi dengan cara mengalihkan *request* berpotensi tersebut ke server palsu *Honeypot* dengan beberapa langkah verifikasi yang dilakukan sebelumnya untuk memastikan apakah *request* yang dikirimkan berpotensi serangan atau tidak.

Selain terlihat dari yang dihasilkan oleh *ipvs_administration*, hasil pengujian tersebut bisa dilihat dari histori atau *log* yang terdapat pada Sistem Deteksi Intrusi dibawah ini.

```
Every 2.0s: tail /var/log/snort/honey.log

192.168.1.253
192.168.1.251
192.168.1.252
192.168.1.254
```

Gambar 14 Hasil Serangan DDOS Pada Sistem Deteksi Intrusi

Pada gambar 14, terlihat jelas bahwa alamat IP yang digunakan oleh komputer penyerang untuk melakukan serangan DDOS terekam seluruhnya oleh Sistem Deteksi Intrusi. Artinya seluruh alamat IP tersebut akan dimasukkan kedalam daftar *blacklist* untuk dijadikan referensi pemblokiran alamat IP yang menuju ke komputer server target nantinya.

III. Simpulan

Berdasarkan hasil pengujian yang dilakukan pada sistem yang dibangun dalam penelitian ini, maka dapat didapatkan beberapa kesimpulan sebagai berikut:

1. Dengan mengkolaborasikan beberapa komponen sistem yang dibangun meliputi *LVS Director*, *Sistem Deteksi Intrusi*, dan *Honeypot* mampu mereduksi serangan *DOS* maupun *DDOS* dengan jenis *SYN Flooding*.
2. Dari hasil pengujian performa pada sistem yang dibangun, terdapat peningkatan waktu respon sebesar 5,91% dan penurunan *failed request* sebesar 0,97% jika dibandingkan dengan LVS Director standar.
3. Sistem yang dibangun memiliki *reply rate* yang lebih baik dan jumlah *error* lebih kecil daripada hanya dengan menggunakan *iptables* ketika *request rate* client rendah (< 700 requests/detik). Jika *request rate* lebih besar dari 700 request/s, performa dari sistem yang dibangun berada di bawah LVS yang menggunakan *iptables*. Hal ini dapat dilihat dari rata-rata *reply rate*, *failed requests* dan waktu respon yang dihasilkan.

Saran diberikan dengan tujuan agar para peneliti lain yang tertarik dengan topik penelitian ini mempunyai gambaran yang jelas bagaimana penelitian ini dapat dikembangkan lebih lanjut, yaitu:

1. Penggunaan metode penjadwalan proses yang digunakan pada *LVS Director* yang mempunyai karakteristik yang sama dengan penjadwalan proses yang digunakan pada penelitian ini yaitu *Round Robin*.
2. Penggunaan fitur *Load Balancing* pada *LVS Director* untuk menghindari terjadinya *failover* pada *LVS Director*, sehingga LVS Director tetap *available* menangani request yang masuk baik yang tanpa maupun yang berpotensi serangan DOS maupun DDOS.

IV. Daftar Pustaka

[1] Cassen. A., *Linux Virtual Server High Availability usingVRRPv2*, Linux Virtual Server OpenSource Project

[2] Honeynet, http://www.honeynet.org/project, waktu akses september 2014.

[3] I. P. Noven, Djanali. S, Husni. M (2014), *Verifikasi Signature Pada Kolaborasi Sistem Deteksi Intrusi Jaringan Tersebar Dengan Honeypot*, Tesis Magister, Institut Teknologi Sepuluh Nopember, Surabaya.

[4] Junaidi, A (2010), *Perancangan Kolaborasi Peer-to-peer Sistem Deteksi Intrusi Jaringan Tersebar Dengan Metode Alert Correlation*, Prosiding SNPS X, Institut Teknologi Sepuluh Nopember, Surabaya.

[5] Kargl. F, Maier. J, Weber, M. (2001), *Protecting web servers from distributed denial of service attacks*, Proceedings of the 10th international conference on World Wide Web.

[6] Kuwatly. I, Sraj. M, Al Masri. Z, Artail. H. (2004), *A Dynamic Honeypot Design for Intrusion Detection*, Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference, pp. 95- 104.

[7] Mirkovic. J, Arikan. E, W. Songjie, Fahmy. S, Thomas. R, Reiher. P. (2006), *Benchmarks for DDoS Defense Evaluation*, Military Communications Conference, 2006. MILCOM 2006. IEEE, pp.1-10, 23-25 Oct. 2006.

[8] O'Rourke. P, Keefe. M. (2001), *Performance Evaluation of Linux Virtual Server*, LISA 2001 15th System Administration Conference, pp 79–92.

[9] Peng. T., Leckie. C., dan Ramamohanara., K. (2007), *Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems*, ACM Comp. Surv. 39, 1, Article 3.

[10] Sardana. A, Kumar. K, Joshi. R. (2007), *Detection and Honeypot Based Redirection to Counter DDoS Attacks in ISP Domain*, Information Assurance and Security, 2007. IAS 2007. Third International Symposium, pp.191-196, 29-31 Aug. 2007.

[11] Spitzner, Lance (2003), *Honeypots: Tracking Hackers*, Pearson Education, Boston.

Halaman ini sengaja dikosongkan.