JUTI: Jurnal Ilmiah Teknologi Informasi - Volume 12, Nomor 2, Juli 2014: 70 – 82

VERIFIKASI SIGNATURE PADA KOLABORASI SISTEM DETEKSI INTRUSI JARINGAN TERSEBAR DENGAN HONEYPOT

Noven Indra Prasetya[1)], Supeno Djanali[2)], Muchammad Husni[3)]
[1)] Program Studi Teknik Informatika, Universitas Wijaya Kusuma Surabaya
[2, 3)] Jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember
e-mail: noven_ip@yahoo.co.id[1)], supeno@its.ac.id[2)], mhusni@gmail.com[3)]

ABSTRAK

Sistem Deteksi Intrusi atau IDS merupakan suatu sistem yang banyak digunakan oleh para administrator jaringan untuk mendeteksi adanya usaha-usaha penyusupan secara proaktif berdasarkan signature atau anomali dengan cara mengeluarkan alert (peringatan) sebagai hasil dari pendeteksiannya. Namun dalam implementasinya, IDS terkadang mengeluarkan alert yang salah (true negative) atau tidak dapat mendeteksi intrusi yang bersifat Zero-Day attack, sehingga menyebabkan administrator kesulitan dalam menganalisa serangan dan tindakan yang tepat selanjutnya. Untuk menyelesaikan permasalahan yang terjadi, maka perlu dilakukan suatu kolaborasi antar IDS maupun dengan teknologi keamanan jaringan yang lain dengan tujuan untuk membuat pendeteksian intrusi menjadi lebih baik.

Pada penelitian ini, diajukan suatu bentuk Kolaborasi IDS dengan teknologi keamanan jaringan, yaitu dengan Honeypot. Selain itu diajukan pula suatu mekanisme perbaikan signature hasil analisis Honeypot dengan tujuan agar signature bisa langsung digunakan oleh IDS tanpa terjadi kesalahan. Pengujian pada lima belas skenario serangan yang dilakukan menunjukkan bahwa konsep kolaborasi dan mekanisme perbaikan signature yang diajukan mampu mereduksi alert true negative dan zero-day attack sebesar 85%. Karena dari lima belas skenario serangan hanya dua skenario serangan yang tidak berhasil terdeteksi oleh IDS dan Honeypot.

***Kata Kunci**: Honeypot, Sistem Deteksi Intrusi, True Negative, Zero-Day Attack*

ABSTRACT

Intrusion Detection System or IDS is a system which is widely used by network administrators to detect any intrusion attempts proactively based on signatures or anomaly by issuing an alert (warning) as a result of the detection. However, implementation of IDS sometimes release false alerts (true negative) or not be able to detect the intrusion that is Zero-Day attack, thus causing difficulty for administrators to analyze attack and to make further appropriate action. To resolve this problem, it is necessary to make a collaboration between the IDS and other network security technologies to make a better intrusion detection system.

In this research, the collaboration is developed between IDS and honeypot. In addition, we also propose a mechanism to improve the signature as a result of the Honeypot analysis; so that, it can be directly used by the IDS without an error. Tests on fifteen attack scenarios show that the concept of collaboration and improvement can reduce true negative alerts and zero-day attacks by 85%; that is, from fifteen attack scenarios only two of them are not successfully detected by the IDS and Honeypot.

***Keywords**: Honeypot, Intrusion Detection System, True Negative, Zero-Day Attack*

I. PENDAHULUAN

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 [4].

Sebagai contoh usaha yang dilakukan para administrator jaringan dalam mengamankan sistem jaringan komputer yang dikelola dari kejahatan *cyber* adalah dengan cara menggunakan teknologi pendeteksi penyusupan yang disebut *Intrusion Detection System* (IDS) atau yang dikenal juga

dengan nama Sistem Deteksi Intrusi. IDS merupakan sebuah sistem yang mempunyai kemampuan untuk monitoring *traffic* jaringan, mendeteksi aktivitas-aktivitas yang mencurigakan, serta mampu melakukan pencegahan dini terhadap intrusi ataupun aktivitas-aktivitas yang dapat membahayakan sistem jaringan komputer yang telah dibangun berdasarkan *signature* atau *anomaly* [7]. Namun dalam implementasinya, IDS terkadang mengeluarkan *alert* yang salah (*true negative*) dan tidak dapat mendeteksi intrusi yang bersifat *Zero-Day attack* atau intrusi yang belum pernah diketahui sebelumnya dan tidak terdapat dalam *signature* (*ruleset*) IDS, sehingga menyebabkan administrator kesulitan dalam menganalisa serangan dan tindakan yang tepat selanjutnya [5].

Untuk menyelesaikan permasalahan yang dimiliki oleh IDS, maka kolaborasi antar IDS, IDS dengan operasi manajemen jaringan, IDS dengan mekanisme keamanan jaringan (*firewall*), IDS dengan teknologi keamanan jaringan yang lain sangat disarankan oleh para praktisi dan peneliti [11]. Kolaborasi IDS bertujuan untuk membuat pendeteksian intrusi menjadi lebih baik, akurat, efisien, dan lebih mudah diatur.

Pada penelitian ini, IDS akan dikolaborasikan dengan teknologi keamanan jaringan Honeypot dimana Honeypot merupakan suatu sistem yang dibuat menyerupai sistem aslinya yang bertugas untuk mendeteksi dan menganalisis *request* yang masuk [8]. Namun pada kenyataan hasil analisis yang dihasilkan oleh Honeypot tidak bisa langsung bisa digunakan sebagai *signature* atau *ruleset* baru untuk IDS dikarenakan terdapat beberapa parameter dari hasil analisis Honeypot yang tidak bisa dikenali oleh IDS. Oleh karena itu, pada penelitian ini juga akan ditambahkan sebuah sistem perbaikan *signature* atau *ruleset* hasil analisis Honeypot dengan tujuan agar *signature* bisa langsung digunakan oleh masing-masing IDS tanpa terjadi kesalahan. Sehingga diharapkan informasi *alert* yang dihasilkan oleh kolaborasi ini semakin valid dan mampu mereduksi alert true negative dan intrusi yang bersifat *Zero-Day attack* dengan lebih baik.

II. SISTEM DETEKSI INTRUSI

Sistem Deteksi Intrusi atau IDS merupakan sebuah sistem yang mempunyai kemampuan untuk monitoring *traffic* jaringan, mendeteksi aktivitas-aktivitas yang mencurigakan, serta mampu melakukan pencegahan dini terhadap intrusi ataupun aktivitas-aktivitas yang dapat membahayakan sistem jaringan komputer yang telah dibangun. Beberapa keuntungan yang diperoleh dengan menggunakan IDS adalah sebagai berikut [7]:

1. Sistem Deteksi Intrusi dapat memberikan informasi apabila terdapat *worm* yang menyerang atau apabila sistem komputer telah berhasil dipenetrasi.
2. Sistem Deteksi Intrusi mampu memonitor jaringan internal terhadap perilaku yang melanggar kebijakan sebuah institusi.
3. Sistem Deteksi Intrusi dapat membantu pelacakan setelah terjadinya suatu serangan, apa saja yang telah diserang dan dari mana serangan tersebut berasal.

Sistem Deteksi Intrusi mampu menyediakan informasi sejauh mana *firewall* bekerja dan siapa saja yang berhasil lolos melewatinya.

A. Klasifikasi Sistem Deteksi Intrusi

Secara umum, Sistem Deteksi Intrusi dapat dibagi berdasarkan klasifikasi sesuai dengan kemampuan yang dimilikinya, yaitu *Detection Method*, dan *Audit Source* [10].

1) Detection Method

Detection Method merupakan pengklasifikasian terhadap penggunaan secara umum pada Sistem Deteksi Intrusi. Terdapat dua jenis Detection Method yang terdapat pada Sistem Deteksi Intrusi, yaitu:

1. Misuse Detection

Misuse Detection atau biasa juga disebut dengan *Signature Analysis* merupakan metode deteksi berdasarkan pada pengumpulan *Attack Signature* (tanda-tanda serangan) yang nantinya akan disimpan ke dalam *database* [3]. Apabila suatu ketika terjadi serangan dan

bentuk dari serangan tersebut terdapat pada *database*, maka Sistem Deteksi Intrusi dapat mengambil suatu *action* atau tindakan terhadap serangan yang terdeteksi.

2. *Anomaly Detection*
   Anomaly Detection atau biasa juga disebut dengan *Behavior Approach* merupakan metode deteksi berdasarkan pada gambaran statistik dari perilaku user atau aplikasi dengan tujuan untuk mendeteksi aksi-aksi yang tidak normal yang dilakukan oleh user atau aplikasi [3].

2) Audit Source

Audit Source merupakan pengklasifikasian Sistem Deteksi Intrusi berdasarkan pemeriksaan data yang diproses. Terdapat dua kategori yang digunakan untuk pemeriksaan data yang diprosesnya, yaitu:

1. *Host-based Intrusion Detection System* (HIDS)
   Host-based Intrusion Detection System atau HIDS merupakan jenis Sistem Deteksi Intrusi yang diimplementasikan langsung ke dalam suatu sistem yang sedang diproteksi untuk dimonitor aktivitas sistem internalnya [7]. HIDS berfungsi untuk memantau dan menghadang *system call* yang dicurigai untuk mencegah terjadinya intrusi terhadap *host*, oleh karena itu HIDS umumnya di-binding dengan *kernel* dan *service* sistem operasi [10]. Selain itu, kemampuan lain yang bisa dilakukan oleh HIDS adalah dapat memantau aliran data dan aktivitas pada aplikasi tertentu, misalnya HIDS yang digunakan untuk mencegah intrusi pada web server pada suatu jaringan.
2. *Network-based Intrusion Detection System* (NIDS)
   Network-based Intrusion Detection System atau NIDS merupakan jenis Sistem Deteksi Intrusi yang tidak melakukan monitoring secara khusus di satu komputer *host* saja, tetapi melakukan monitoring dan proteksi dalam satu jaringan secara global [7]. Teknologi NIDS yang berkembang sekarang ini tidak hanya berfungsi sebagai monitoring jaringan, tapi bisa juga digunakan untuk mencegah ataupun *filtering* terhadap *request-request* yang masuk menuju jaringan internal. Karena NIDS yang ada sekarang ini menggabungkan fitur Sistem Deteksi Intrusi dengan *firewall*, maka terkadang disebut sebagai *In-Line* IDS atau *Gateway Intrusion Detection System* (GIDS).

B. Macam-Macam Alert Sistem Deteksi Intrusi

Alert merupakan salah satu bentuk aksi yang dihasilkan oleh Sistem Deteksi Intrusi apabila terdapat *request* yang dianggap berpotensi serangan. *Alert* pada Sistem Deteksi Intrusi memiliki tingkatan tertentu untuk menunjukkan ke-validan suatu *request* yang ditujukan kepada suatu sistem jaringan komputer. Macam-macam atau tingkatan *alert* pada Sistem Deteksi Intrusi sebagai berikut [5]:

1. *True Positive*
   Suatu kondisi dimana Sistem Deteksi Intrusi mendeteksi request yang masuk merupakan suatu bentuk penyerangan, sehingga Sistem Deteksi Intrusi mengeluarkan *alert*. Dan ternyata *request* tersebut betul-betul merupakan suatu bentuk penyerangan terhadap jaringan internal.
2. *True Negative*
   Suatu kondisi dimana Sistem Deteksi Intrusi mendeteksi *request* yang masuk bukan merupakan suatu bentuk penyerangan, sehingga Sistem Deteksi Intrusi tidak mengeluarkan *alert*. Namun ternyata *request* tersebut merupakan suatu bentuk penyerangan terhadap jaringan internal.
3. *False Positive*
   Suatu kondisi dimana Sistem Deteksi Intrusi mendeteksi *request* yang masuk merupakan suatu bentuk penyerangan, sehingga Sistem Deteksi Intrusi mengeluarkan *alert*. Namun ternyata *request* tersebut bukan merupakan suatu bentuk penyerangan terhadap jaringan internal.

4. *False Negative*
   Suatu kondisi dimana Sistem Deteksi Intrusi mendeteksi *request* yang masuk bukan merupakan suatu bentuk penyerangan, sehingga Sistem Deteksi Intrusi tidak mengeluarkan *alert*. Dan ternyata *request* tersebut betul-betul bukan merupakan suatu bentuk penyerangan terhadap jaringan internal.

III. KOLABORASI SISTEM DETEKSI INTRUSI

Untuk mendeteksi dan mencegah adanya aktifitas yang mencurigakan pada suatu jaringan serta menghindari dari kemungkinan adanya perusakan dalam suatu sistem atau jaringan, maka kolaborasi antar Sistem Deteksi Intrusi (IDS), IDS dengan operasi manajemen jaringan, IDS dengan mekanisme keamanan jaringan (seperti *Firewall*), IDS dengan teknologi keamanan jaringan yang lain sangat disarankan oleh para praktisi dan peneliti [11]. Dengan adanya kolaborasi Sistem Deteksi Intrusi maka kelemahan yang terdapat pada Sistem Deteksi Intrusi akan dapat diminimalisasi [4]. Karena Kolaborasi Sistem Deteksi Intrusi bertujuan untuk membuat pendeteksian intrusi menjadi lebih baik, akurat, efisien, dan lebih mudah diatur dengan cara menggabungkan informasi dari gabungan antar IDS maupun dengan teknologi keamanan jaringan yang lain.

IV. SNORT

Snort merupakan *software* paket *sniffer* dan *logger open source* yang dibangun berdasarkan *library* libpcap, juga merupakan Sistem Deteksi Intrusi standar Unix's *family* baik dalam bentuk Sistem Deteksi Intrusi berbasis *Host* (HIDS) maupun Sistem Deteksi Intrusi berbasis Jaringan (NIDS) . Sehingga Snort dapat digunakan untuk mendeteksi usaha penyusupan yang ditujukan untuk jaringan maupun komputer atau *host* [7].

A. *Snort Signatures*

Signature adalah suatu *pattern* yang dapat dilihat pada sebuah data *packet*. Sebuah *signature* dapat digunakan untuk mendeteksi satu atau banyak jenis serangan. Sebagai contoh, "scripts/iisadmin" adalah sebuah paket yang ditujukan pada web server yang dapat mengindikasikan sebuah aktifitas intrusi.

B. *Snort Alerts*

Alert adalah salah satu bentuk notifikasi dari suatu aktifitas intrusi. Ketika Snort mendeteksi seorang *intruder*, maka Snort akan menginformasikan hal ini kepada administrator dengan menggunakan *alert*. *Alert* dapat dijumpai dalam bentuk *pop-up windows* dan e-mail. Alert juga disimpan dalam *log files* dan *database* agar dapat dianalisa lebih lanjut.

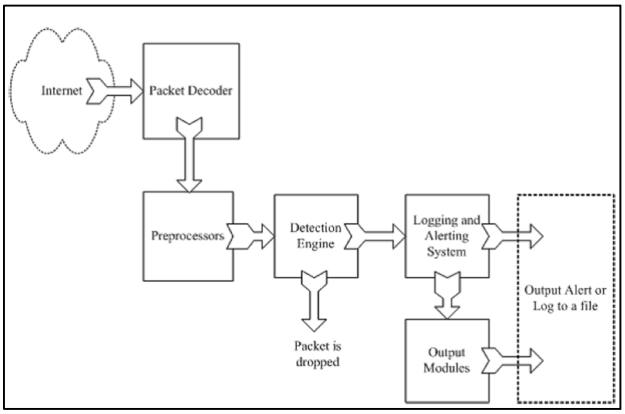
C. *Snort Logs*

Secara *default*, Snort menyimpan file-file log nya pada /var/log/Snort. Log ini disimpan dengan format *text* dan format *binary*.

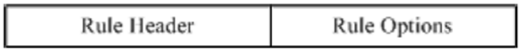
D. *Komponen Snort*

Snort dibagi menjadi lima komponen utama yang masing-masing komponennya saling bekerja sama untuk mendeteksi suatu serangan tertentu. Komponen-komponen tersebut antara lain:

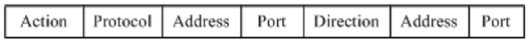
1. Packet Decoder
   Packet decoder berfungsi untuk mengambil packet dari tipe-tipe *network interface* yang berbeda dan menyiapkan packet tersebut untuk di-*preprocess* atau dikirim ke *detection engine*. *Interface* dapat berupa Ethernet, SLIP, dan PPP.



Gambar 1. Komponen Snort



Gambar 2. Struktur Rule Snort



Gambar 3. Struktur Rule Header

```
alert icmp any any -> any any (msg:
"Ping with TTL=100";\
ttl: 100;)
```

Gambar 4. Contoh Rule Snort

2. Preprocessors
   Preprocessors adalah komponen atau *plug-in* yang dapat digunakan dengan Snort untuk menyusun atau memodifikasi *packet* sebelum sampai ke *detection engine*. Beberapa *preprocessors* juga dapat melakukan pendeteksian dengan cara menemukan kejanggalan-kejanggalan pada *packet header* dan kemudian men-*genenerate alert*. *Preprocessor* juga digunakan untuk melakukan *packet defragmentation*. Ketika data yang berukuran besar ditransfer ke suatu *host*, *packet* akan difragmentasi. Sebagai contoh, panjang packet maksimum pada *ethernet network* adalah 1500 *bytes*, yang berarti jika kita mengirim data yang berukuran lebih dari 1500 *bytes*, maka data itu akan dipecah-pecah menjadi beberapa data *packet* sehingga tiap ukuran *packet* adalah kurang dari 1500 *bytes*. Sistem penerimaan memiliki kemampuan untuk *re-assemby data packets* yang diterima untuk kemudian membentuk *data packet* yang sebenarnya.
3. *Detection Engine*
   Detection engine merupakan bagian terpenting dari *Snort*. Fungsi dari komponen ini adalah untuk mendeteksi jika ada aktifitas intrusi pada sebuah packet. Detection engine menggunakan Snort rules untuk menjalankan fungsinya. Snort rules dibaca ke struktur data internal dimana mereka akan mengalami kecocokan dengan *packets*. Jika sebuah *packet* mengalami kecocokan dengan sebuah *rule*, maka *detection engine* akan mengeluarkan *alert*.
   Jika *traffic* pada sebuah jaringan terlalu padat, Snort kemungkinan besar tidak dapat melakukan deteksi secara *real-time*. Beberapa faktor yang mempengaruhi performa *detection engine* adalah jumlah *rules*, kecepatan *bus* dan *traffic* pada jaringan.
   Proses *logging packet* atau *generating alert* sangat bergantung pada *rule*. Sebuah *low priority rule* akan men-*generate* sebuah *low priority alert* dan begitu juga sebaliknya. Perlu diingat bahwa *detection engine* dapat meng-*apply rules* pada bagian yang berbeda-beda pada *packet*. Bagian-bagian ini antara lain adalah IP *header*, *Transport layer header*, *application layer header* dan *packet payload.*
4. *Logging* dan *Alerting system*
   Bergantung pada apa yang ditemukan oleh *detection engine* di dalam packet, *packet* dapat digunakan untuk melakukan *logging* atau men-*generate alert*. Logs disimpan di dalam sebuah *text* file yang sederhana, format tcp-dump atau format lainnya. Semua file log disimpan pada direktori /var/log/Snort.

E. Struktur Rule Snort

Semua *Snort rules* memiliki dua bagian *logical*, yaitu *rule header* dan *rule option.* Hal ini dapat dilihat pada Gambar 2.

Rule header berisi informasi mengenai action apa saja yang terdapat pada *rule*. *Rule header* juga berisi kriteria-kriteria untuk mencocokan *rule* dengan *packets*. *Rule options* berisi sebuah pesan *alert* dan informasi mengenai bagian mana dari *packet* yang seharusnya digunakan untuk meng-*generate* pesan *alert*. *Rule options* juga berisi kriteria tambahan untuk mencocokan rule dengan *packets*. Sebuah rule dapat mendeteksi satu atau lebih aktifitas intrusi. Pada Gambar 3 ditunjukkan struktur umum untuk *rule header*.

Bagian *action* dari *rule header* menentukan tipe tindakan apakah yang akan diambil jika kriteria telah cocok dan *rule* juga telah cocok dengan *packet*. Tindakan yang umumnya diambil adalah melakukan *generate alert* atau pesan log. Bagian *protocol* dari *rule header* digunakan untuk meng-*apply rule* pada *packet* untuk sebuah protokol tertentu saja. Ini adalah kriteria pertama yang disebutkan di dalam *rule*. Beberapa contoh protokol yang digunakan adalah IP, ICMP dan UDP. Bagian *address* dari *rule header* menentukan alamat sumber dan alamat tujuan. Alamat dapat berupa sebuah *host* atau banyak *host*. Bagian ini juga dapat digunakan untuk meng-*exclude* beberapa alamat dari suatu jaringan. *Direction* pada *rule header* menentukan alamat dan *port* mana yang digunakan sebagai sumber dan tujuan. Sebagai contoh, *rule* pada Gambar 4 akan meng-*generate alert* setiap mendeteksi ICMP *packet* dengan TTL bernilai 100.

Bagian dari *rule* sebelum tanda kurung adalah *rule header*, sedangkan bagian *rule* di dalam kurung adalah bagian dari *options*. *Header* terdiri dari bagian-bagian berikut ini:

1. *Rule Action*
   Pada *rule* ini tindakan yang diambil adalah "*alert*" yang berarti bahwa *alert* akan di-*generate* ketika kondisi terpenuhi.
2. *Protocol*
   Pada rule ini protokol yang dipilih adalah ICMP yang berarti bahwa *rule* tersebut hanya bisa di-*apply* pada packet ICMP.
3. *Source Address* dan *Source Port*
   Pada rule ini diisi dengan "*any*" yang berarti *rule* dapat di-*apply* pada semua *packet* yang berasal dari semua *address*.
4. *Destination Address* dan *Destinaton Port*
   Pada rule ini diisi dengan "*any*" yang berarti *rule* dapat di-*apply* pada semua *packet* yang menuju ke semua *address*.

V. 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* [8]. 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, di bawah ini akan dijelaskan beberapa alasan menggunakan Honeypot [8].

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 [8]. 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 [2].
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 [8]. 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 [2].

C. Fungsi Honeypot

Berdasarkan bentuk penanganan yang dilakukan terhadap *intruder*¸ Honeypot memiliki fungsi sebagai berikut [1]:

1. *Prevention*
   Yang dimaksud *prevention* di sini adalah menggunakan Honeypot sebagai pengalih

perhatian untuk mencegah terjadinya serangan. Konsep utama yang terdapat pada area ini adalah bagaimana membuat penyerang menghabiskan waktu dan sumber daya yang dimiliki untuk menyerang Honeypot dan bukannya menyerang sistem produksi.

2. *Detection*
   Honeypot berguna pada proses pendeteksian karena dapat menyederhanakan proses pendeteksian. Besarnya *traffic* jaringan yang ada pada saat ini dapat menimbulkan beban yang berlebihan pada sumber daya keamanan yang ada. *Traffic* yang besar tersebut harus dipilah satu persatu oleh sumber daya keamanan. Dikarenakan Honeypot tidak memiliki aktivitas produksi, maka setiap koneksi dari dan menuju Honeypot langsung dapat dicurigai. Secara umum, setiap koneksi yang berhubungan dengan Honeypot dapat dikategorikan sebagai aktivitas yang tidak terotorisasi. Hal ini memudahkan di dalam pencarian dan pengumpulan informasi yang diinginkan.

VI. HONEYTRAP

Honeytrap adalah sebuah *Low Interaction Honeypot* yang berfungsi untuk mengobservasi serangan terhadap servis jaringan. Berbeda dengan honeypot lainnya, yang lebih fokus pada pengumpulan *malware*, Honeytrap bertujuan untuk menangkap usaha ekploitasi lebih dini, dengan cara mengumpulkan usaha eksploitasi tersebut dan selanjutnya melakukan *tracing* terhadap proses serangan [2].

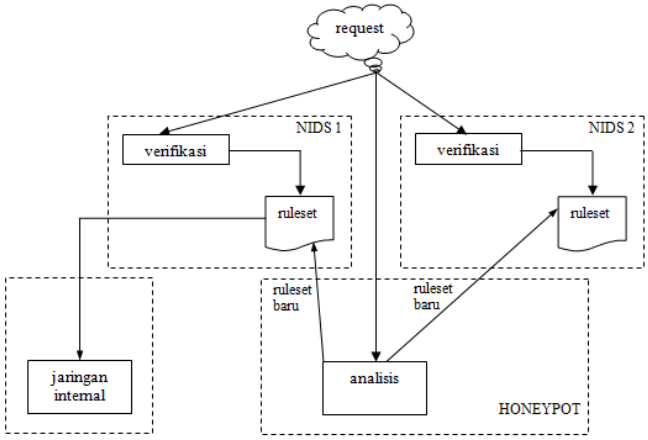
Honeytrap mampu memproses serangan yang belum diketahui sebelumnya, walaupun Honeytrap belum diberikan *pre-knowledge* tentang protokol dan *vulnerability* yang berguna untuk dirinya dalam melakukan pendeteksian nantinya. *Incoming data* pada Honeytrap di-*handle* sebagai *byte stream per session.* Maka *stream* ini dapat diproses dengan modul-modul yang berbeda, sebagai contoh adalah *decoding* dan *virus scanning*. Honeytrap bekerja dengan cara mengekstrak sebuah koneksi yang masuk dengan menggunakan *stream monitors.* Jika sebuah monitor mendeteksi sebuah request menuju *unbound port*, maka proses pada server yang akan menanganinya.

A. *Service Emulation*

Honeytrap memiliki kemampuan untuk mengemulasikan *service*. Honeytrap dapat membaca respon *default* untuk *port* yang spesifik yang berasal dari file yang berisi *captured message* dari *service* yang sebenarnya.

B. *Nebula*

Nebula adalah *plug-in* pada Honeytrap yang berguna untuk melakukan analisa terhadap serangan menuju Honeytrap dan kemudian men-*generate signature* Snort dari serangan tersebut. Nebula memiliki server yang independen, dimana Honeytrap sebagai *client* harus melakukan autentikasi



Gambar 5. Desain Kolaborasi Sistem Deteksi Intrusi Dengan Honeypot

setiap kali mengirim *attacks string* ke server Nebula. Performa proses *generating signature* Snort pada Nebula sangat bergantung pada *value threshold* dan *similarity* yang digunakan sebagai acuan dalam men-*generate signature*.

VII. LANGKAH-LANGKAH PENELITIAN

Langkah-langkah penelitian merupakan penerapan yang dilakukan setelah melalui tahapan rancangan kegiatan penelitian. Pada sub bab ini akan dijelaskan deskripsi umum sistem, arsitektur sistem, dan implementasi sistem yang dibuat pada penelitian ini.

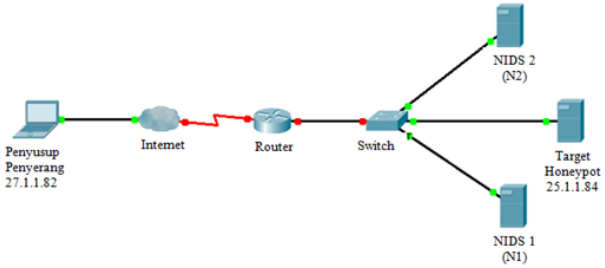
A. Deskripsi Umum Sistem

Berdasarkan penjelasan yang dijelaskan pada bab kajian pustaka dan dasar teori, selanjutnya akan dibuat suatu model desain kolaborasi Sistem Deteksi Intrusi dengan Honeypot. Kolaborasi Sistem Deteksi Intrusi pada ini terdiri dari beberapa sensor dan sistem Perbaikan Signature, dimana sensor yang dimaksud berupa dua buah Sistem Deteksi Intrusi berbasis jaringan (NIDS) dan Honeypot. Sedangkan sistem Perbaikan Signature yang dimaksud adalah sebuah *software* aplikasi yang dibuat dengan tujuan untuk memperbaiki *Signature* hasil dari analisis Honeypot supaya bisa digunakan sebagai *ruleset* baru pada semua Sistem Deteksi Intrusi berbasis jaringan tanpa terjadi kesalahan atau *error*.

Pada kolaborasi ini, Honeypot bekerja bersama dengan Sistem Deteksi Intrusi berbasis jaringan untuk melakukan *monitoring request* yang masuk ke dalam jaringan internal. Bedanya, Sistem Deteksi Intrusi berbasis Jaringan bekerja melakukan *monitoring request* yang masuk ke sejumlah subnet jaringan internal yang dilindungi oleh IDS, sedangkan Honeypot bertugas melakukan monitoring *request* ke dalam komputer server atau target tempat Honeypot tersebut berada, atau dengan kata lain Honeypot bekerja seperti konsep Sistem Deteksi Intrusi berbasis Host (HIDS), yaitu melakukan *monitoring request* yang masuk hanya pada komputer server atau target penyusupan.

Semua *request* yang masuk akan dimonitoring oleh semua sensor tak terkecuali Honeypot. Hasil monitoring masing-masing IDS akan dikirimkan ke komputer server untuk dikumpulkan menjadi satu dan ditampilkan dalam bentuk antar muka berbasis web dengan tujuan memudahkan administrator jaringan untuk memantau aktivitas jaringan yang dikelola. Sedangkan hasil monitoring Honeypot akan disimpan untuk selanjutnya dibandingkan atau dianalisis dengan *request* lain yang berhasil direkam oleh Honeypot. Jika terdapat dua jenis *request* yang sama atau memiliki nilai *similarity* sebesar 10%, maka Honeypot akan menyimpulkan kalau kedua *request* tersebut merupakan usaha penyusupan kepada server dan Honeypot men-*generate* hasil analisis tersebut menjadi *signature* yang akan digunakan sebagai perbaikan *ruleset* untuk masing-masing Sistem Deteksi Intrusi.

Selanjutnya hasil analisis yang dihasilkan oleh Honeypot tersebut secara berkala akan didistribusikan ke sejumlah Sistem Deteksi Intrusi berbasis Jaringan sebagai ruleset baru baik yang bersifat *whitelist* maupun *blacklist*. Sehingga diharapkan *alert false positive* dan *zero-day attack* yang merupakan kelemahan pendeteksian Sistem Deteksi Intrusi dapat diminimalisasi. Dengan cara



Gambar 6. Arsitektur Sistem

tersebut diharapkan administrator jaringan dapat dengan mudah mengambil tindakan selanjutnya setelah melihat hasil yang diberikan oleh sistem.

B. Arsitektur Sistem

Arsitektur sistem yang digunakan pada penelitian ini terdiri dari beberapa komputer yang bertindak sebagai penyusup, internet, router, NIDS, dan server/target/Honeypot yang secara keseluruhan di-virtualisasikan dalam lingkungan *virtual machine* VMware dan ditunjukkan pada Gambar 6. Komputer penyusup adalah komputer yang di dalamnya terdapat *software-software exploit* yang ter-*include* pada sistem operasi Linux Backtrack, berfungsi untuk melakukan usaha-usaha penyusupan terhadap komputer server atau target. Komputer internet adalah sebuah komputer yang digunakan seolah-olah sebagai jaringan internet yang berfungsi untuk menghubungkan antara komputer penyusup dengan jaringan internal target penyusupan. Komputer Router adalah komputer yang bertugas untuk mengarahkan semua *request* yang masuk ke komputer server atau target. Komputer NIDS 1 dan 2 adalah komputer yang di dalamnya terinstal *software* Snort sebagai *engine signature base* NIDS, berfungsi memonitoring semua *request* yang masuk ke dalam jaringan internal baik *request* yang benar ataupun *request* yang berpotensi penyusupan.

Komputer server atau target terdiri dari paket *software* Prelude-IDS yang berfungsi sebagai antar muka berbasis web, *software* Honeytrap sebagai *engine low interaction* Honeypot dan *plugin* Nebula sebagai *engine* yang berfungsi untuk men-*generate signature rulebase* IDS.

VIII. HASIL UJI COBA DAN PEMBAHASAN

Pada bab hasil uji coba dan pembahasan ini akan dijelaskan pengujian terhadap sistem yang dibangun dari segi fungsionalitas dan kinerja terhadap beberapa jenis serangan untuk diketahui seberapa baik kinerja yang diberikan oleh sistem dalam menangani beberapa jenis serangan yang dilakukan.

A. Lingkungan Pengujian

Pengujian pada penelitian ini dilakukan pada lingkungan virtual yang diemulasikan pada komputer portabel dengan menggunakan *software virtual machine* VMware. Topologi pengujian sesuai dengan arsitektur sistem yang terdapat pada bab metode penelitian dengan spesifikasi perangkat keras dan perangkat lunak yang sudah ditentukan untuk mendukung penelitian ini berjalan dengan baik. Berikut spesifikasi perangkat keras dan perangkat lunak yang digunakan selama melakukan penelitian hingga tahap pengujian.

1. Server / Honeypot
   a. IP Address 25.1.1.84
   b. RAM 512 MB
   c. HD 20 GB
   d. OS Linux Ubuntu 10.04 LTS
   e. Aplikasi : prelude-ids 0.9, honeytrap 1.1.0, nebula 0.3, mysql-server 5.1, openjdk 6, cron 3.0
2. NIDS1
   a. IP Address 25.1.1.82
   b. RAM 256 MB
   c. HD 10 GB
   d. OS Linux Ubuntu 10.04 LTS
   e. Aplikasi : snort 2.8, prelude-ids 0.9, cron 3.0
3. NIDS2
   a. IP Address 25.1.1.83
   b. RAM 256 MB

   c. HD 10 GB
   d. OS Linux Ubuntu 10.04 LTS
   e. Aplikasi : snort 2.8, prelude-ids 0.9, cron 3.0
4. Router
   a. IP Address eth0 26.1.1.82, eth1 25.1.1.83
   b. RAM 64 MB
   c. HD 10 GB
   d. OS Linux Ubuntu Hardy Heron
   e. Aplikasi : iptables 1.3.8
5. Internet
   a. IP Address eth0 27.1.1.81, eth1 26.1.1.81
   b. RAM 64 MB
   c. HD 10 GB
   d. OS Linux Ubuntu Hardy Heron
6. Client
   a. IP Address 27.1.1.82
   b. RAM 256 MB
   c. HD 20 GB
   d. OS Linux Backtrack R5
   e. Aplikasi : nmap 6.01, hydra 7.3, httpdx 0.8, guildftpd 0.x.x

B. Pengujian Sistem

Pengujian sistem dilakukan untuk mengetahui kemampuan sistem dalam menangani beberapa skenario serangan yang dibuat. Skenario serangan yang dibuat terdiri dari teknik serangan yang digunakan dan protokol tujuan serangan. Tabel I merupakan detail dari skenario penyerangan yang dibuat untuk membuktikan pengujian sistem dari sisi kinerja. Berdasarkan pengujian yang dilakukan pada kelima belas skenario yang dibuat dan sudah dibuktikan secara kinerja pada masing-masing skenario, maka dapat disimpulkan pada Tabel II.

Pada Tabel II terdapat simbol "√" yang berarti berhasil terdeteksi, sedangkan simbol "—" berarti gagal terdeteksi. Jadi sebagai contoh, usaha serangan *port scanning* berhasil terdeteksi oleh masing-masing Sistem Deteksi Intrusi yaitu NIDS1 dan NIDS2 dan menghasilkan *alert* atas usaha serangan tersebut. Kemudian Honeypot pada usaha *port scanning* tidak berhasil mendeteksi dan mengeluarkan *alert*, sehingga secara otomatis Kolaborasi pada usaha port scanning bertanda "—" yang berarti tidak ada *generate* dan distribusi *signature* baru oleh Honeypot kepada masing-masing

TABEL I. SKENARIO PENGUJIAN KINERJA SISTEM

Usaha Serangan	Software atau Exploit	Protokol Tujuan
Port Scanning	nmap	-
Remote Brute Force	hydra	http
Arbitrary Remote Delete Files	guildftpd	ftp
Arbitrary Create Directorie & File	httpdx	http
Remote Buffer Overflow	wu-ftpd remote bof	ftp
Windows Exploit	msfconsole	netbios
Mercury Remote Buffer Overflow	mercury-cram-md5	imap
Web Server Exploit	sws_connection_bof	http-ssl
Telnet Brute Force	medusa	vnc
Getting Shell With SMB	msfcli	smb
TCP SYN Flood	hping3	netbios-ssn
TCP Connection Flood	hping3 with sarfu	microsoft-ds
UDP Flood	udp.pl exploit	dns
ICMP Flood	letdown	microsoft-ds
Metasploit Browser Autopwn	msfconsole	http

Sistem Deteksi Intrusi.

Berdasarkan data pengujian yang terdapat pada Tabel II hasil pengujian kinerja sistem kolaborasi, dapat diketahui bahwa kolaborasi Sistem Deteksi Intrusi dengan Honeypot yang dilakukan pada penelitian ini dapat bekerja dengan baik sesuai dengan konsep kolaborasi Sistem Deteksi Intrusi yang berkembang. Dari kelima belas skenario pengujian yang dilakukan, beberapa penilaian pada masing-masing skenario dianggap berhasil sesuai dengan konsep kolaborasi Sistem Deteksi Intrusi yaitu masing-masing Sistem Deteksi Intrusi dan Honeypot mampu bekerja sama dalam mendeteksi usaha penyerangan yang dilakukan oleh attacker dan menghasilkan alert atas usaha penyerangan tersebut. Hasil deteksi yang dihasilkan oleh Honeypot berupa *signature* baru yang selanjutnya secara otomatis akan didistribusikan ke sejumlah Sistem Deteksi Intrusi sebagai *ruleset* baru bagi Sistem Deteksi Intrusi dengan tujuan agar kedua Sistem Deteksi Intrusi memiliki kemampuan yang sama dalam mendeteksi usaha-usaha penyerangan yang masuk ke dalam jaringan yang dilindungi oleh Sistem Deteksi Intrusi selanjutnya menghasilkan sejumlah *alert* yang sesuai dengan kriteria yang terdapat pada usaha penyerangan tersebut.

Dari data hasil pengujian pada Tabel II dapat disimpulkan bahwa Kolaborasi Sistem Deteksi Intrusi dengan Honeypot yang dilakukan pada penelitian ini mampu mereduksi atau mengurangi adanya *alert true negative* dan *zero-day attack* pada Sistem Deteksi Intrusi. Dari lima belas usaha penyerangan yang dilakukan selama pengujian, hanya terdapat dua usaha penyerangan yang sama sekali tidak berhasil terdeteksi oleh Sistem Deteksi Intrusi dan Honeypot, sedangkan tiga belas usaha penyerangan mampu terdeteksi oleh masing-masing Sistem Deteksi Intrusi secara bergantian sesuai dengan *ruleset* yang tertanam pada masing-masing Sistem Deteksi Intrusi. Bahkan dari tiga belas usaha penyerangan, enam usaha penyerangan yang berhasil di-*generate* oleh Honeypot menjadi *signature* baru yang selanjutnya didistribusikan ke sejumlah Sistem Deteksi Intrusi.

IX. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian yang dilakukan pada kolaborasi Sistem Deteksi Intrusi, mekanisme perbaikan *signature ids*, dan pendistribusian *alert* yang dilakukan oleh Honeypot, maka dapat diperoleh beberapa kesimpulan sebagai berikut:

TABEL II. HASIL PENGUJIAN KINERJA SISTEM KOLABORASI

Usaha Serangan	NIDS1	NIDS2	Honeypot	Kolaborasi
Port Scanning	√	√	–	–
Remote Brute Force	–	–	√	√
Arbitrary Remote Delete Files	–	√	√	√
Arbitrary Create Directorie & File	–	√	√	√
Remote Buffer Overflow	–	√	√	√
Windows Exploit	–	–	–	–
Mercury Remote Buffer Overflow	√	–	–	–
Web Server Exploit	–	–	√	√
VNC Server Buffer Overflow	–	√	√	√
Getting Shell With SMB	√	–	–	–
TCP SYN Flood	–	√	–	–
TCP Connection Flood	–	√	–	–
UDP Flood	–	√	–	–
ICMP Flood	–	√	–	–
Metasploit Browser Autopwn	–	–	–	–

1. Dengan kemampuan yang dimiliki oleh Honeypot sebagai sistem deteksi intrusi *hostbased*, maka Honeypot bisa dikolaborasikan dengan beberapa Sistem Deteksi Intrusi yang diimplementasikan secara tersebar di dalam jaringan sesuai dengan konsep Kolaborasi Sistem Deteksi Intrusi.
2. Mekanisme perbaikan dan pengiriman *alert* yang dilakukan oleh Honeypot bertujuan untuk mendistribusikan hasil korelasi kesemua sensor sebagai perbaikan *ruleset* baru yang dimiliki oleh Sistem Deteksi Intrusi dengan harapan pendeteksian berikutnya lebih akurat, menghasilkan alert yang benar atau *true positive*, dan meminimalisasi *zero day attack*.

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 Honeypot hadir dengan berbagai macam aplikasi dengan kemampuan dan karekteristik yang beraneka ragam. Sehingga sangat memungkinkan apabila konsep kolaborasi antara Sistem Deteksi Intrusi dengan Honeypot pada penelitian ini dilakukan dengan menggunakan perangkat lunak Honeypot yang lain.

DAFTAR PUSTAKA

[1] A.P., Baskoro (2011), "Pengalihan Paket Ke Honeypot Pada Linux Virtual Server Untuk Mengatasi Serangan DDOS", Magister, Institut Teknologi Sepuluh Nopember, Surabaya.
[2] Honeytrap, http://honeytrap.carnivore.it/, waktu akses september 2011.
[3] Hwang, K (2004), "Cooperative Anomaly and Intrusion Detection for Alert Correlation in Networked Computing Systems", IEEE Transaction on Dependable and Secure Computing, Oakland, CA.
[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] Kruegel, C., Robertson, W., Vigna, G (2004), "Using Alert Verification to Identify Successful Intrusion Attempts", Practice in Information Processing and Communication (PIK), Vol.27, No.4 , pp. 219-227.
[6] Nebula, http://nebula.carnivore.it/, waktu akses september 2011.
[7] Scott, C., Wolfe, P., Hayes, B (2004), Snort for Dummies, Wiley Publishing, Hoboken.
[8] Spitzner, Lance (2003), Honeypots: Tracking Hackers, Pearson Education, Boston.
[9] Werner, T., Fuchs, C., Gerhards, E., Martini, P. (2009), "Nebula - Generating Syntactical Network Intrusion Signatures", University of Bonn, Germany.
[10] Winarno, Idris (2008), "Pengembangan Kolaborasi Sistem Deteksi Intrusi Jaringan Tersebar Berbasis Publish-Subscribe Dengan Menggunakan Metode Alert Correlation", Magister, Institut Teknologi Sepuluh Nopember, Surabaya.