BAB II

Tinjauan Pustaka

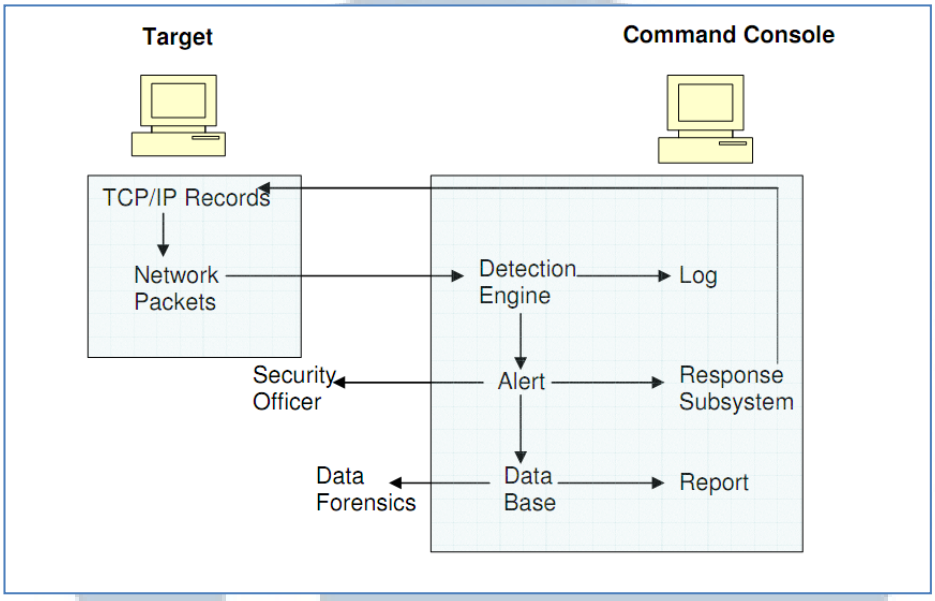
2.1 *Intrusion Detection System*

2.1.1 Pengertian IDS

Intrusion Detection System adalah sebuah sistem yang digunakan untuk memonitor jaringan dari serangan atau penyusupan dan melaporkan temuan tersebut ke *network administrator* agar bisa ditindaklanjuti [6]. Sistem tersebut membaca tiap paket yang ditransfer dalam sebuah jaringan. Jika terdapat paket yang berbahaya sesuai *rules* yang sudah ditetapkan, maka akan dilakukan beberapa tindakan, diantaranya pelaporan, penyimpanan data, dan analisis data.

2.1.2 Snort

Snort [2] adalah salah satu IDS yang mendapat pengakuan sebagai standar *de facto* IDS [16]. Snort adalah *packet sniffer* berbasis *libpcap* yang digunakan sebagai *network intrusion detection system* (NIDS) yang ringan [17]. Snort menyimpan *log* dari *pattern matching* konten dan mendeteksi serangan, seperti *buffer overflows, stealth port scans*, CGI *attacks*, SMB *probes*, dan lain-lain. Snort memiliki kemampuan untuk memberikan *alert* dengan berbagai cara. Seperti mencatat *alert* ke sys*log*, *Server Message Block* "WinPopup" messages, atau sebuah *file alert* yang terpisah.



Gambar 2.1 Skema IDS [6]

Skema tersebut menggambarkan bahwa konsol IDS membaca tiap *network packet* dari target. Paket tersebut kemudian dideteksi apakah paket tersebut merupakan paket *malicious*. Setiap hasil pendeteksian akan dimasukkan ke *log*. Jika terdapat temuan adanya paket *malicious*, konsol akan mengirimkan sinyal *alert* yang akan dikirimkan ke petugas/admin dan menyimpan temuan tersebut ke database yang akan digunakan untuk membuat laporan.

2.1.3 Arsitektur Snort

Untuk memonitor sebuah jaringan, Snort memiliki tiga subsistem utama dalam arsitektur Snort: *packet decoder*, mesin pendeteksi, sistem *log* dan *alert*.

2.1.3.1 *Packet Decoder*

Packet Decoder bertugas untuk mendekode paket-paket yang ditransfer dalam jaringan tersebut. *Decoder* tersebut diatur dalam data-link *layer*. Tiap

subrutin dalam *decoder* membubuhkan urutan pada paket data. Subrutin tersebut akan dipanggil sesuai dengan urutan *protocol*, dari *data link layer* sampai *transport layer*. Snort mempunyai kemampuan *decode* untuk *Ethernet*, SLIP, dan *raw* (PPP) *data-link protocols*.

2.1.3.2 Mesin Deteksi

Mesin deteksi Snort menggunakan *linked list* dua dimensi yang disebut sebagai *chained header* dan *chained option*. *Chain header* berisi informasi berupa atribut umum dari serangan yang ditemukan. Sedangkan *chain option* berisi *option modifier* dari serangan tersebut. Sistem ini digunakan untuk memudahkan proses pencarian dan *query* temuan. Contohnya, jika ada 50 serangan dari sumber dan ke sasaran yang sama, maka seranngan tersebut akan disimpan dalam satu *chained header* dengan 50 *chained option*.

```
        Figure 3 - Rule Chain logical structure
-------------------------------------------------------------------------------
 ------------------------            ------------------------            -----
| Chain Header           |          | Chain Header           |          | Chai
|                        |          |                        |          |
| Source IP Address      |          | Source IP Address      |          | Sour
| Destination IP Address |--------->| Destination IP Address |--------->| Dest
| Source Port            |          | Source Port            |          | Sour
| Destination Port       |          | Destination Port       |          | Dest
|                        |          |                        |          |
 ------------------------            ------------------------            -----
            |                                  |
            |                                  |
            |                                  |
           \|/                                \|/
 -----------V---------              -----------V---------
| Chain Option        |             | Chain Option        |
|                     |             |                     |
| Content             |
| TCP Flags           |
| ICMP Codes/types    |
| Payload Size        |
| etc.                |
|                     |
 ---------------------
            |
            |
            |
           \|/
 -----------V---------
| Chain Option        |
|                     |
| Content             |
| TCP Flags           |
| ICMP Codes/types    |
| Payload Size        |
| etc.                |
|                     |
 ---------------------
            |
            |
```

Gambar 2.2 *Double Linked List* Mesin Deteksi [18]

2.1.3.3 Sistem *Log* dan *Alert*

Saat ini ada tiga pilihan pencatatan *log* yang disediakan oleh Snort. Ketiga pilihan tersebut adalah paket *log* yang dienkripsi, dalam format yang dapat dibaca oleh manusia, atau kedalam format tcpdump (http://www.tcpdump.org). *Log* yang dienkripsi memiliki keunggulan dalam membaca data, sedangkan format tcpdump dapat disimpan ke *hard disk* dengan lebih cepat. Snort bisa memberi *alert* kedalam lima bentuk. Bentuk-bentuk *alert* tersebut adalah dikirim ke sys*log*, dituliskan dalam dua tipe *plain text*, dan dikirim berupa Winpopup.

2.1.4 Komponen Pendukung Snort

Snort hanya menyediakan tampilan berupa *command line*. Untuk medapatkan tampilan statistik yang lebih baik, pengguna diharuskan menginstall aplikasi *reporting*. Aplikasi *open source* yang banyak dipakai adalah BASE, Snorby, dan SqueRT [3]. Dari ketiga aplikasi di atas, aplikasi yang memiliki fitur paling banyak dalam bidang pelaporan adalah BASE [3].

BASE adalah akronim dari "*Basic Analysis and Security Engine*". BASE dibuat berdasarkan *Analysis Console for Intrusion Databases* (ACID). BASE menyediakan tampilan statistik sebuah halaman web. BASE juga menyediakan grafik dari serangan-serangan yang ditemukan [4].

2.2 *Instant Messenger*

2.2.1 Pengertian *Instant Messenger*

Instant messenger adalah sebuah komunikasi *real-time* yang menyajikan komunikasi berbasis teks, audio, dan video. *Instant Messenger* juga memungkinkan penggunanya untuk bertukar data dalam bentuk *file*.

2.2.2 *Instant Messenger* sebagai *Notification*

Instant Messenger mulai banyak digunakan sebagai *Notification* oleh *web system* [9]. IM bisa menyediakan informasi tentang keadaan sistemnya. Selain itu,

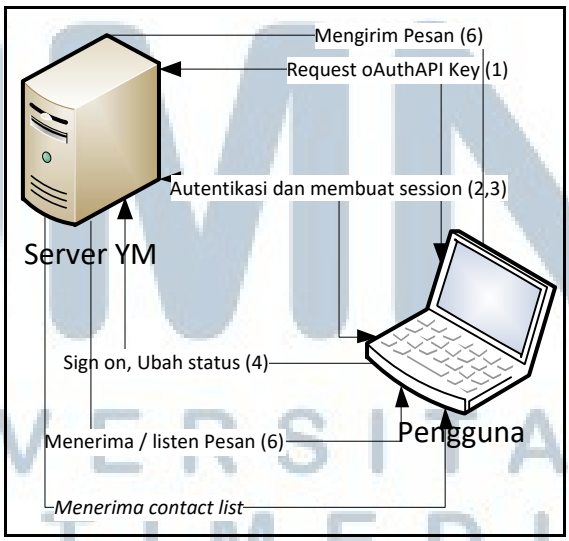
admin juga bisa menggali informasi yang dibutuhkan dengan menggunakan *query* ke sistem melalui *instant messenger*.

2.2.3 Yahoo! Messenger API

Yahoo! Messenger menyediakan API (*Application Program Interface*) untuk memungkinkan *developer* membangun aplikasi yang terintegrasi dengan Yahoo! Messenger [19]. Untuk berkomunikasi dengan Yahoo! Messenger, pengguna diharuskan untuk melakukan otorisasi untuk melakukan kegiatan dengan Yahoo! Messenger. Setelah itu, pengguna akan diberi *token* yang memiliki batas waktu sebagai identifikasi.

2.2.4 Mekanisme Yahoo! Messenger API

Tahapan-tahapan yang dilakukan ketika berkomunikasi dengan menggunakan Yahoo! Messenger API [19] dapat dilihat pada gambar 2.3.



Gambar 2.3 Proses Yahoo! Messenger API

1. Membuat Yahoo! Messenger *Open Authentication (OAuth) API Key*.

Yahoo! Messenger *Open Authentication Key* dibuat hanya sekali. *Key* inilah yang nantinya kan digunakan setiap aplikasi/sistem disambungkan ke *server* Yahoo! Messenger. Satu aplikasi hanya membutuhkan satu *key* dan akan diberikan ketika mendaftarkan aplikasi tersebut ke Yahoo! Messenger SDK. Bersamaan dengan *Oauth key*, juga akan diberikan sebuah *secret key* yang akan digunakan ketika akan membuat sebuah koneksi ke *server* Yahoo! Messenger.

2. Autentikasi dengan server Yahoo! Messenger IM SDK.

Proses ini terdiri dari beberapa langkah yang secara garis besar bertukar *key* untuk meyakinkan bahwa aplikasi yang digunakan sudah terdaftar di Yahoo! Messenger SDK. Pertama-tama, aplikasi meminta sebuah *request token*. Saat meminta *request token,* aplikasi juga menyertakan *Oauth key* yang sudah didapat ketika mendaftarkan aplikasi tersebut. Setelah mendapat *request token*, aplikasi harus mengirim sebuah jawaban yang mengirimkan *request token* tersebut disertai *Oauth key* dan *Signature key* yang merupakan modifikasi dari *secret key*. Setelah itu, *server* akan memberikan jawaban berupa *access token* yang akan digunakan pada tahap-tahap selanjutnya.

3. Membuat *session* baru.

Session baru dibuat dengan cara mengirimkan *access token* dan informasi tentang autentikasi lainnya. *Session* inilah yang akan digunakan pada saat

mengirim/menerima pesan. Proses inilah yang sebenarnya terjadi ketika pengguna menekan tombol *sign in* pada aplikasi Yahoo! Messenger.

4. Menerima dan membarui informasi sekarang.

Setelah mendapat *session*, pengguna dapat menggunakan fitur-fitur seperti yang ada pada Yahoo! Messenger. Di antaranya adalah menerima dan membarui informasi. Informasi tentang pengguna pada akun Yahoo! dapat diubah setelah pengguna *sign in*.

5. Menerima *contact list* dan detail-detail terkait dengan *contact*.

Selain itu, pengguna juga bisa menerima dan memanipulasi kontak yang ada pada akunnya. Fitur inilah yang memungkinkan *programmer* dapat membuat aplikasi yang menampilkan kontak yang ada di daftar kontak.

6. Mengirim dan menerima pesan dari *contact* yang lain.

Fitur ini adalah fitur utama dalam API Yahoo! Messenger. Untuk mengirim pesan, aplikasi hanya tinggal mengirimkan detail pesan dan penerima ke server Yahoo! Messenger. Dengan detail itu, Yahoo! akan mengirimkan pesan ke tujuan.

Untuk menerima pesan, proses yang dilakukan adalah aplikasi menjalankan sebuah *listener* yang akan mengambil paket berisi pesan teks atau notifikasi lainnya yang sudah diterima. Paket inilah yang akan diproses untuk mengetahui lebih lanjut apakah paket itu berisi pesan atau notifikasi lainnya.

Notifikasi lainnya di sini dapat berupa permintaan teman dan lain sebagainya. Proses *listener* ini juga akan mendapatkan paket yang diterima ketika pengguna sedang berada dalam jaringan (*on line*).

2.3 *Command Line Interface*

Command Line Interface adalah sebuah antar muka yang berupa komunikasi antara komputer dan manusia, atau antara dua program, dimana kedua pelaku komunikasi tersebut bertukar informasi berupa sebaris teks (*command line*) [20]. Perintah-perintah yang digunakan dalam CLI disimpan dalam sebuah *file* dan digunakan oleh sebuah interpreter untuk mengartikan perintah tersebut agar bisa dimengerti komputer. Beberapa antar muka grafis (*Graphical User Interface*) menggunakan CLI untuk memanggil program-program pendukung untuk membuka *file* atau program.

2.3.1 Penggunaan CLI

Antar muka *command line* digunakan bilamana program menyangkut perbendaharaan kata dan *query* dalam jumlah besar dan mempunyai pilihan yang besar pula. Hal seperti itu biasa terjadi pada sebuah sistem operasi . Selain itu, CLI juga digunakan ketika sebuah *environment* tidak mendukung untuk menggunakan GUI.

CLI banyak digunakan oleh *system administrator* maupun *programmer* pada lingkungan teknik maupun akademik. Selain itu, CLI juga dapat digunakan

untuk tunanetra karena baris perintah dapat ditampilkan dalam huruf braille menggunakan *refreshable Braille displays*.

2.3.2 Keuntungan CLI

Command Line Interface memiliki keunggulan dibandingkan dengan *Graphical User Interface* [20]. Keunggulan tersebut diantaranya cepat dan *powerful* apabila digunakan oleh orang yang berpengalaman; interaksi dikendalikan oleh pengguna; interaksi dengan komputer minim (tanpa *mouse*); dan dapat digabungkan dengan gaya antarmuka yang lain [20].

Dibalik kelebihan-kelebihan tersebut, CLI memiliki beberapa kelemahan [20] seperti hanya sedikit *prompting* yang dilakukan sehingga tidak cocok untuk pengguna baru; membutuhkan pengalaman dan pengetahuan pengguna akan sistem atau program; mengandalkan ingatan pengguna akan perintah dan sintaks; relatif susah untuk dipelajari; dan rentan *error*.

2.3.3 *Parsing*

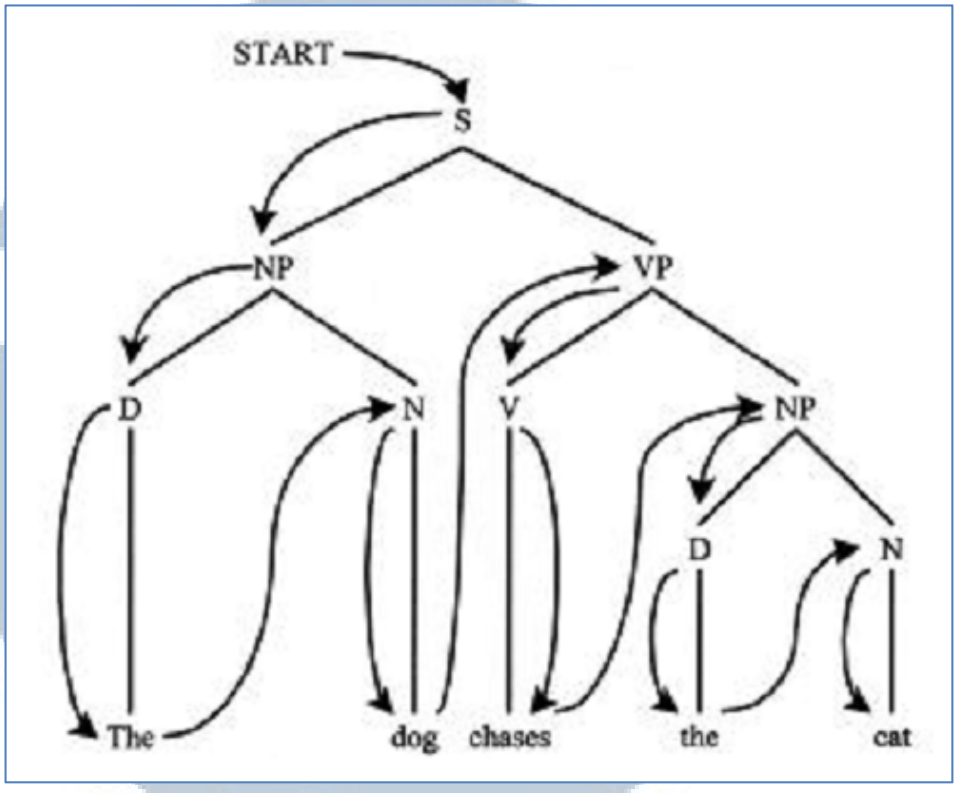
Parsing adalah proses yang dilakukan oleh *interpreter* untuk menerjemahkan masukan ke bahasa mesin. Tidak hanya dilakukan pada CLI, metode *parsing* juga digunakan pada *compiler* bahasa pemrograman untuk mendapatkan arti dari sebuah sintaks. *Parsing* sendiri bertujuan untuk memilah-milah kalimat untuk kemudian dibagi menjadi beberapa *constituent*. *Constituent* adalah unsur pembentuk kalimat yang bisa berdiri sendiri, seperti *noun phrase* dan *verb phrase*.

Metode *Parsing* dapat dilakukan dalam dua cara, yaitu *top down* dan *bottom up* [21].

2.3.3.1 *Top Down Parsing*

Top Down Parser bekerja dengan cara menguraikan sebuah kalimat mulai dari c*onstituent* yang terbesar sampai menjadi *constituent* yang terkecil. Hal ini dilakukan terus-menerus sampai semua komponen yang dihasilkan adalah *constituent* terkecil dalam kalimat, yaitu kata.

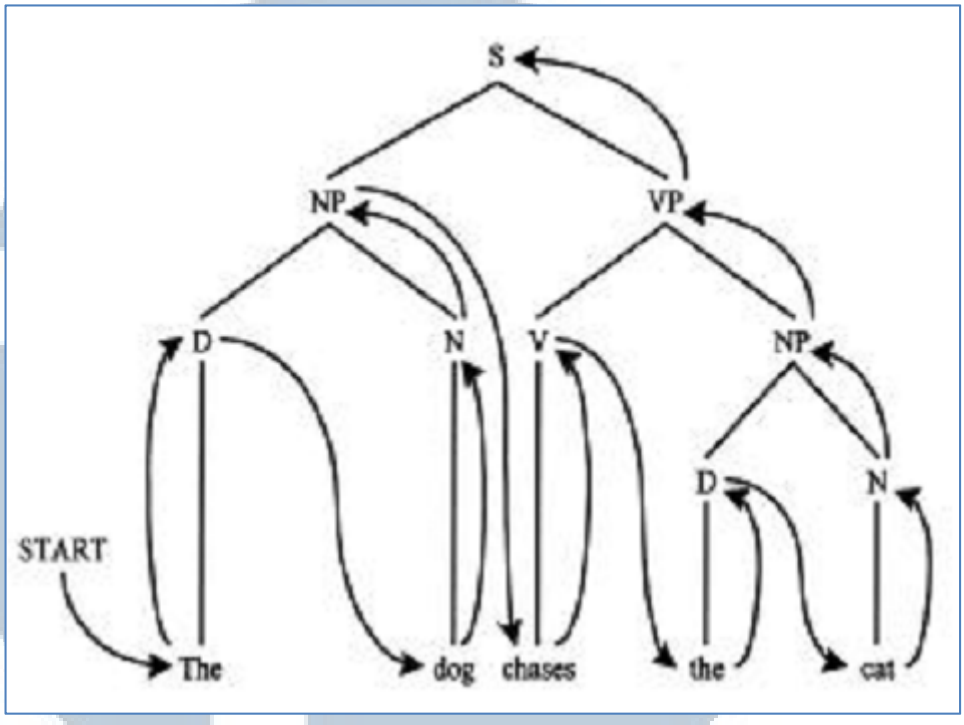
Seperti terlihat pada gambar 2.4, metode ini memecah sebuah kalimat "*The dog chases the cat*". Kalimat itu dipecahkan kata per kata berdasarkan konstituennya. Jadi kalimat itu diartikan dengan cara memecah dari bagian terbesar sampai terkecil.



Gambar 2.4 Metode *Top Down Parsing* [21]

2.3.3.2 *Bottom-Up Parsing*

Kebalikan dari metode sebelumnya, *bottom-up parser* bekerja dengan cara mengambil satu demi satu kata yang diberikan untuk dirangkai menjadi *constituent* yang lebih besar. Hal ini dilakukan secara terus menerus sampai *constituent* yang didapatkan berupa sebuah kalimat. Seperti pada gambar 2.5, kalimat tersebut diartikan kata per kata hingga semua kata diartikan sebagai sebuah kalimat secara utuh.



Gambar 2.5 Metode *Bottom Up Parsing* [21]

Salah satu penerapan metode *bottom up parsing* adalah dengan menggunakan metode

2.3.4 CLI *Guideline*

Menurut Kiewe [22] dalam situs konsultasi pembuatan antarmuka *command line*, ada beberapa hal yang harus diperhatikan dalam pembuatan *command line interface*. Aturan-aturan tersebut adalah:

1. Struktur bahasa

Hal-hal yang perlu diperhatikan dalam menentukan struktur perintah yang digunakan adalah kesederhanaan kata, penggunaan kembali kata, dan penentuan aturan-aturan struktur yang akan digunakan. Kesederhanaan kata di sini memiliki

makna bahwa semakin sedikit jumlah kata yang digunakan semakin baik, karena semakin sedikit pula kata yang harus diingat. Namun, semakin sedikit banyak jumlah kata berarti juga semakin banyak kata yang harus diketik. Oleh karena itu diperlukan keseimbangan antara kesederhanaan kata dan banyaknya kata yang harus diingat. Misalnya untuk merubah sebuah nama akun pengguna, sebuah CLI mempunyai perintah `edit-user`, sedangkan CLI lainnya diharuskan menggunakan perintah `delete-user` dan `add-user`. CLI pertama mengharuskan pengguna mengingat lebih banyak daripada CLI kedua, namun CLI pertama menyediakan perintah yang mempersingkat proses CLI kedua.

Selain kesederhanaan kata, perlu juga diperhatikan penggunaan kembali sebuah kata. Hal ini berarti satu kata dapat dipakai dalam beberapa perintah. Seperti contoh untuk menambah akun pengguna bisa menggunakan perintah `new` dan untuk menambah *directory* juga menggunakan perintah `new`.

2. Pemilihan kata

Sebaiknya CLI menggunakan Bahasa Inggris dalam pemilihan kata dibandingkan dengan simbol. Contohnya penggunaan kata `remove` dibandingkan dengan rm. Kata yang digunakan juga layaknya yang sering digunakan. Struktur yang digunakan sebaiknya berupa predikat+objek.