BAB II
LANDASAN TEORI

2.1 Snort

Snort merupakan aplikasi atau perangkat lunak berbasis *opensource* yang memiliki keunggulan untuk mengetahui adanya indikasi penyusupan pada jaringan berbasis TCP/IP secara *real time* (Mutaqin, 2016). Jika terindikasi adanya penyusupan, Snort akan melakukan pencatatan atau *logging* terhadap paket-paket yang telah terdeteksi sebagai intrusi berdasarkan aturan yang telah ditetapkan.

Pada implementasinya, Snort memiliki aturan yang telah dikonfigurasi untuk mendeteksi intrusi dalam sebuah jaringan. Terdapat sebuah *database* yang mencakup aturan-aturan yang memiliki fungsi tertentu sehingga dapat digunakan sesuai dengan kebutuhan. Setiap paket yang lalu lintas jaringan akan dianalisa dengan cara melakukan pencocokkan terhadap aturan yang telah ditetapkan. Hasilnya, Snort akan melakukan *logging* ke dalam *database* seperti *MySQL* maupun *file log* terhadap paket yang terindikasi sebagai sebuah intrusi.

2.1.1 Mode Snort

Dalam pemahaman lain, Snort dapat didefinisikan sebagai *single-threaded* yang artinya hanya dapat mengeksekusi satu tugas dalam satu waktu yang mana dapat berjalan pada empat mode (Rani & Singh, 2012). Keempat mode operasional tersebut yaitu :

a. Packet Sniffer Mode

Pada modus operasi ini bertugas untuk menangkap paket-paket pada lalu lintas jaringan serta menampilkan dalam bentuk aliran yang bersifat *continuous* pada sebuah layar. Paket-paket dalam lalu lintas jaringan akan ditangkap secara *real time*. Dalam penerapannya, *packet sniffer mode* ini bekerja seperti aplikasi *TCPdump* yang umum digunakan untuk menangkap paket lalu lintas jaringan.

Ketika snort dijalankan, *packet sniffer mode* berjalan dengan melakukan *listen* yang kemudian menangkap paket-paket yang terdapat dalam lalu lintas jaringan. Modus ini menghasilkan ringkasan paket yang diambil termasuk protokol, statistik fragmentasi paket, dan statistik aliran.

b. Packet Logger Mode

Umumnya, modus operasi ini mencatat *log* dari paket-paket pada lalu lintas jaringan dan menyimpannya ke dalam *disk*. Ketika mengetahui adanya paket, Snort juga dapat mencatat paket tersebut ke *file log*. *File log* dapat dilihat menggunakan Snort maupun *TCPdump*. Tidak ada aktivitas deteksi intrusi yang dilakukan oleh Snort dalam mode operasi ini.

Ketika berjalan dalam mekanisme ini, Snort mengumpulkan setiap paket yang ditangkap dan menyimpannya pada *file log* dalam direktori Snort yang tersusun secara hirarki. Dengan kata lain, sebuah *file* baru akan dibuat untuk setiap aktifitas yang ditangkap dan informasi sesuai dengan waktu terjadinya insiden.

c. Detection Mode

Pada modus ini, Snort akan menangkap paket-paket lalu lintas jaringan dan menganalisanya untuk dibandingkan dengan aturan yang sudah ditetapkan oleh Administrator. Selain itu, modus ini juga melakukan beberapa tindakan berdasarkan sesuatu yang telah teridentifikasi.

Mekanisme dari *detection mode* yaitu menggunakan konfigurasi dari aturan yang telah ditetapkan oleh pengguna. Aturan tersebut terdapat dalam *file* snort.conf yang merupakan *file* utama untuk menentukan apakah akan dilakukan tindakan tertentu atau tidak. Konfigurasi snort.conf merupakan *file* yang otomatis dibuat ketika aplikasi Snort terpasang pada sistem. Namun, konfigurasi dilakukan secara manual oleh pengguna.

d. Inline Mode

Pada *inline mode*, Snort memperoleh paket dari *IP table* bukan *library* libpcap dan kemudian menggunakan jenis aturan yang baru untuk membantu *IP table* mengijinkan atau menghentikan paket berdasarkan aturan Snort. Secara *default* Snort berjalan sebagai IDS, tetapi pada *inline*

mode menunjukkan bahwa Snort selain berfungsi sebagai IDS, aplikasi tersebut juga mampu menjadi *Intrusion Prevention System* (IPS). Dengan kata lain, pada pengaturan awal Snort berada pada *passive mode* yang mana hanya memberikan peringatan mengani adanya intrusi.

2.1.2 Komponen Snort

Keempat modus operasional Snort berkaitan dengan mekanisme deteksi yang dilakukan oleh beberapa komponen Snort. Paket-paket lalu lintas jaringan yang melewati Ethernet dimana Snort terpasang akan melalui beberapa komponen yang bertujuan untuk mendeteksi adanya intrusi. Komponen tersebut yaitu *packet capture-decode engine, preprocessor, detection engine, logging* and *alerting system,* dan *output plugins* yang dapat dilihat pada Gambar 2.1. Komponen-komponen Snort tersebut yaitu:

a. Packet Capture-Decode Engine

Snort mengambil paket-paket data yang melewati Ethernet dengan menggunakan *library* yang dinamakan libpcap. Paket-paket data tersebut dinamakan *frame* dimana paket data itu berasal dari *Layer Data Link*. Paket data yang telah diambil kemudian dikirimkan ke komponen *decode engine* dan dipecah untuk mendapatkan informasi lebih detail.

Komponen ini merupakan komponen awal yang dilalui oleh paket data. Paket data yang masuk akan diterjemahkan dengan membaca kode pada *frame*, protokol IP, dan TCP/UDP. Setelah proses selesai, informasi tersebut kemudian dilanjutkan ke komponen selanjutnya.

b. Preprocessor

Paket yang telah dikirimkan oleh *decode engine* kemudian diinvestigasi terlebih dahulu sebelum dikirim ke *detection engine*. Komponen ini tidak melakukan modifikasi apapun pada paket yang akan dikirimkan ke komponen *detection engine* untuk dilakukan analisa.

Pada dasarnya komponen ini melakukan investigasi paket data berdasarkan *signature-based*. Ketika dilakukan investigasi dimana terdapat serangan yang tidak dapat dideteksi oleh *signature-based* pada *detection*

engine, komponen ini mampu digunakan untuk menemukan serangan berbasis *non-signature*. Preprosesor lainnya bertanggung jawab untuk normalisasi lalu lintas sehingga *detection engine* dapat membandingkan paket dengan *signature-based*.

c. Detection Engine

Komponen *detection engine* merupakan komponen utama yang sangat penting karena komponen ini berfungsi untuk mendeteksi apakah paket tersebut berupa sebuah instrusi atau tidak. Mekanisme komponen ini yaitu dengan mengambil informasi dari paket yang telah dikirimkan oleh komponen sebelumnya (*decode* dan *preprocessor*) yang kemudian membandingkan informasi tersebut dengan aturan-aturan yang telah ditetapkan oleh pengguna.

Dalam mekanismenya, *detection engine* berjalan dengan *signature-based*. Artinya pengguna harus melakukan konfigurasi terlebih dahulu untuk menambah, memodifikasi, maupun menghapus aturan yang akan digunakan. Jika serangan terdeteksi sesuai dengan aturan yang telah ditetapkan, selanjutnya *detection engine* akan mendapatkan informasi yang akan digunakan oleh komponen selanjutnya yaitu *system log* untuk menghasilkan *alert*. Namun, jika paket tidak sesuai dengan aturan maka paket tersebut akan dibuang.

d. Logging and Alerting System

Komponen ini memiliki tanggung jawab untuk menghasilkan peringatan dan pencatatan pesan yang didapatkan dari *detection engine*. Informasi yang ditemukan dalam paket yang terindikasi berupa intrusi akan dicatat pada sebuah *file*. *File* tersebut berupa *log* aktivitas yang akan digunakan untuk menghasilkan peringatan. Semua *file log* ditempatkan pada direktori /var/log/snort yang tersusun secara *default* ketika pemasangan aplikasi Snort.

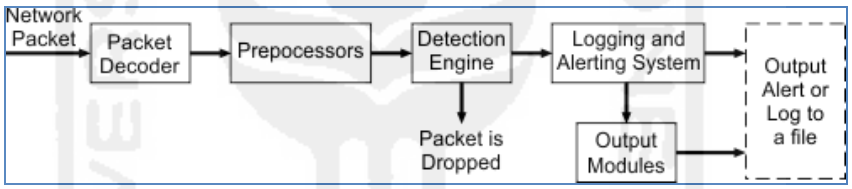
Hasil deteksi tidak hanya dicatat pada sebuah *file* saja, tetapi juga disimpan dalam sebuah *database*. Informasi yang masuk ke dalam *database* akan dipisah sesuai dengan bagian masing-masing. Misalnya mengenai jenis

protokol, alamat IP, jenis serangan, dan informasi lainnya. Hal tersebut berguna untuk mempermudah pengguna dalam membaca aktifitas yang teridentifikasi sebagai intrusi.

e. Output Modules

Berbeda dengan komponen lainnnya, *output modules* dapat melakukan tindakan bagaimana hasil pencatatan dari komponen *logging* and *alerting system* tersebut disimpan. Secara teknis modul ini mengontrol jenis *output* yang dihasilkan.

Pada penelitian ini, *output* yang dihasilkan berupa alarm yang diintegrasikan dengan aplikasi *instant messaging* Telegram secara *real time*. Selain itu, laporan indikasi intrusi juga dapat dilihat pada *interface* berbasis *web*.



Gambar 2.1 Komponen Snort (Kinal & Hajdarevic, 2013)

Untuk mendapatkan hasil yang maksimal terhadap penyimpanan aktifitas yang terindikasi sebagai sebuah intrusi dibutuhkan sebuah aplikasi pendukung yaitu Barnyard2. Barnyard2 merupakan aplikasi perangkat pendukung berbasis *opensource* yang berfungsi untuk menerjemahkan *alert unified* dimana *output* bersifat *binary* dari Snort agar dapat disimpan dalam *database* dan mudah dipahami oleh pengguna. Secara singkat, aplikasi barnyard2 berfungsi sebagai jembatan antara Snort dan *database* sehingga *alert* tersebut dapat dipahami oleh pengguna. Aplikasi tersebut memiliki keunggulan untuk menyimpan data yang diperoleh secara efektif dengan cara mengurangi beban dari sensor *engine detection* utama pada snort.

Pada implementasinya, Snort menggunakan aturan-aturan dengan pola berupa *signature* dinamakan *rule*. Aturan tersebut terletak pada direktori snort /etc/snort/rules dimana akan digunakan untuk mendeteksi sebuah intrusi. Snort

rules terdiri dari dua bagian, yaitu *rule header* dan *rule option* (Rafiudin, 2010). Bagian *rule header m*erupakan bagian yang mengidentifikasi aksi dari sebuah kondisi seperti halnya *alert, log, pass, activate, dynamic,* dan lain-lain. *Rules header* meliputi aksi, protokol, *IP Address* asal dan *IP Address* tujuan beserta *netmask,* dan *port* asal serta *port* tujuan. Sedangkan *rule option* merupakan bagian *rule* yang mengidentifikasikan pesan peringatan dan informasi deteksi intrusi.

2.2 *Intrusion Detection System*

Intrussion Detection System (IDS) merupakan perangkat lunak atau perangkat keras sistem yang secara otomatis melakukan proses pemantauan (*monitoring*) insiden yang terjadi dalam sistem komputer atau jaringan serta menganalisis tanda-tanda adanya masalah terhadap keamanan sistem (Anitha, 2011). Jika terindikasi adanya aktifitas yang mencurigakan terhadap aliran (*traffic*) paket-paket yang keluar dan masuk pada sistem, maka IDS akan merekam aktifitas tersebut.

2.2.1 Klasifikasi IDS

Penerapan IDS dapat dilakukan diberbagai tempat pada suatu jaringan di sebuah instansi atau perusahaan dengan tujuan tercapainya keamanan sistem. IDS sendiri dapat diklasifikasikan menjadi dua jenis yaitu *Host-based Intrusion Detection System* (HIDS) dan *Network-based Intrusion Detection System* (NIDS) (Thomas, 2005). Kedua jenis IDS adalah sebagai berikut :

a. Host-based Instrusion Detection System (HIDS)

IDS tipe ini diterapkan dan beroperasi pada sebuah komputer *server* yang dianggap kritis atau rawan. Dalam pengertian lainnya, HIDS sesuai untuk arsitektur yang berupa *single server* yang memberikan layanan seperti *web server*, *mail server*, maupun layanan lainnya. Tujuan HIDS untuk memantau serta mendeteksi aliran paket-paket yang masuk dan keluar yang terindikasi berbahaya pada host sehingga tipe ini disebut juga *host-based* IDS.

b. Network-based Instrusion Detection System (NIDS)

Pada IDS jenis ini diterapkan dan beroperasi dengan melihat semua lalu lintas aliran yang melewati jaringan sehingga disebut *network-based* IDS. Pada klasifikasi ini semua paket yang keluar maupun masuk pada sebuah jaringan komputer akan terlebih dahulu dianalisa dengan tujuan untuk menemukan adanya percobaan penyusupan ke dalam sistem jaringan. Hal ini efektif untuk menganalisa *traffic* diantara *host* maupun segmen jaringan lokal. Berbeda dengan HIDS, pada jenis ini NIDS akan ditempatkan pada pintu masuk jaringan (*gateway*).

2.2.1 Metode Deteksi

HIDS dan NIDS memiliki tujuan yang sama yaitu sebagai deteksi intrusi terhadap keamanan sistem. Meskipun memiliki perbedaan pada penerapannya, kedua IDS tersebut memiliki beberapa metode dalam mendeteksi adanya intrusi yaitu *Signature-based* IDS dan *Anomaly-based* IDS (Rani & Singh, 2012).

a. *Signature-based* IDS

Pada teknik berbasis *signature*, IDS memeriksa lalu lintas yang sedang berlangsung yang kemudian dibandingkan dengan pola tertentu agar dapat diketahui apakah terjadi serangan.

Cara kerja metode ini sama halnya seperti *antivirus* dimana akan dilakukan perbandingan pada lalu lintas jaringan dengan *database*. Oleh karena itu, metode ini membutuhkan pembaharuan terhadap *database* IDS.

b. *Anomaly-based* IDS

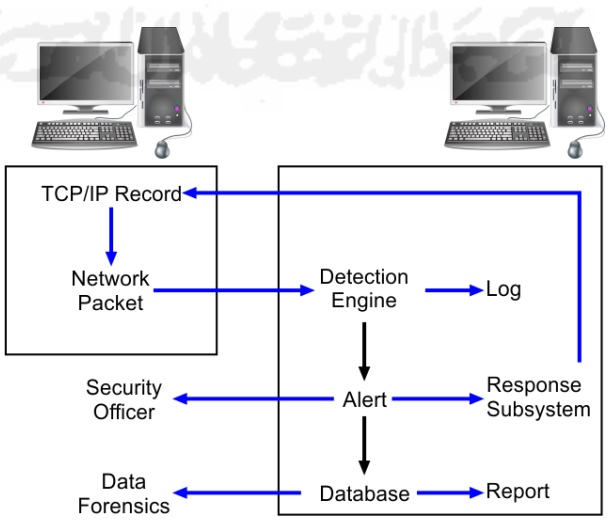
Teknik ini memeriksa pola lalu lintas yang sedang berlangsung pada jaringan atau sistem yang dapat menunjukkan serangan. Secara umum, metode ini membandingkan lalu lintas yang sedang dipantau dengan lalu lintas normal yang biasanya terjadi. Dalam membandingkan keduanya, metode ini menggunakan teknik statistik.

Umumnya IDS menggunakan teknik berbasis *signature* yang membandingkan paket dengan aturan. Hal tersebut sesuai dengan klasifikasi IDS

yang berupa *host-based*. Sedangkan teknik yang berbasis anomali digunakan dalam jaringan yang memiliki skala yang terbilang besar seperti NIDS dengan mengenali anomali lalu lintas jaringan. Dari hasil kedua teknik tersebut, aktifitas intrusi yang terdeteksi akan dicatat dan disimpan dalam *file log* pada sistem dimana *file* tersebut yang digunakan oleh administrator untuk mengetahui insiden yang telah terjadi.

Pada penelitian ini menggunakan teknik yang berbasis *signature* dengan pertimbangan sistem yang dilindungi yaitu sebuah *server* dengan *Operating System* Ubuntu Server yang memiliki beberapa layanan seperti *web server*, FTP *server*, dan SSH. Snort bertugas menjadi pintu masuk (*gateway*) sehingga dapat terlebih dahulu mendeteksi intrusi. Sistem yang terpasang snort terkoneksi dengan *Internet* sehingga paket yang terdeteksi sebagai intrusi akan langsung dikirimkan melalui sebagai alarm melalui aplikasi *instant messaging* Telegram.

Di dalam IDS terdapat sensor yang mampu mendeteksi adanya intrusi. Sebuah sensor digunakan untuk mengenali paket dari jaringan mana paket tersebut datang hingga masuk ke dalam *detection engine* yang akan memicu alarm jika terjadi penyalahgunaan apapun yang terdeteksi. Sensor ini kemudian akan dikirimkan ke berbagai segmen *mission-critical* jaringan. Sebuah *server* digunakan untuk mengumpulkan *file log* maupun alarm dari beberapa sensor. Adapun arsitektur IDS yang dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur IDS (Anitha, 2011)

Pada di atas menunjukkan dimana IDS digunakan untuk mendeteksi serangan dan apabila terdeteksi, maka akan dihasilkan sebuah peringatan atau alarm. Administrator yang menjaga keamanan sistem tersebut akan mendapatkan pemberitahuan tentang penyalahgunaan tersebut misalnya melalui *file log*. Peringatan tersebut disimpan dalam sebuah *database* yang nantinya dapat dianalisis. Hasil peringatan tersebut juga dapat dirangkum menjadi sebuah laporan misalnya dalam bentuk dokumen digital. Laporan dari aktifitas mengenai adanya sebuah intrusi sangat berguna bagi sistem administrator yang mengelola sebuah *server* yang penting. Hal tersebut bisa dijadikan bukti data forensik yang dapat digunakan sebagai acuan untuk perbaikan celah pada sistem. Dengan adanya laporan tersebut, keamanan sistem dapat ditingkatkan serta mengurangi tingkat kerentanan pada sistem.

2.3 *Notification Alert System*

Pendeteksian terhadap penyusupan pada sistem merupakan hal yang penting bagi sistem administrator untuk menjaga keamanan sistem dan data yang terdapat di dalam sistem. Selain untuk keamanan data, pendeteksian tersebut berguna untuk mengetahui pola tindakan yang dilakukan oleh penyusup. *Alert System* yang dibangun merupakan pesan peringatan terhadap sistem berdasarkan hasil laporan yang didapatkan dari proses pengecekan paket-paket lalu lintas jaringan yang kemudian disimpan pada *file log* dan *database*.

Sistem peringatan tersebut merupakan sistem peringatan yang dapat dikirimkan melalui media *email*, SMS, atau media lainnya. Jika terdapat penyusupan pada jaringan berdasarkan laporan yang dihasilkan sebagai akibat dari penilaian menurut IDS, maka peringatan tersebut akan dikirim ke administrator jaringan (Yadav & Mehtre, 2013). Peringatan tersebut bertujuan agar administrator menyadari ancaman yang terjadi pada sistem.

Sistem yang mendeteksi adanya penyusupan kemudian akan melakukan *trigger* untuk melaporkan kepada administrator bahwa sedang terjadi sebuah gangguan pada sistem. *Trigger* tersebut merupakan pemicu dimana ketika terdapat

sebuah paket yang terdeteksi sebagai sebuah intrusi, maka akan dijalankan perintah untuk menghasilkan alarm.

Pemberitahuan alarm mengenai intrusi akan dikirimkan ke administrator melalui media tertentu. Alarm yang dihasilkan akan berjalan secara *real time*, tetapi hanya mengirimkan pemberitahuan yang sangat singkat dalam bentuk *text-based* saat terjadinya sebuah insiden. Sedangkan laporan secara detail akan dikirimkan kepada administrator melalui media seperti *email* atau lainnya yang memuat catatan lengkap keamanan sistem. Informasi yang terdapat dalam laporan tersebut sangat penting karena hal tersebut dapat menentukan titik-titik yang rawan dalam jaringan.

Pada penelitian sejenis lainnya, pemberitahuan mengenai intrusi terhadap sistem memanfaatkan *SMS Gateway*. Umumnya, media SMS memiliki keterbatasan jumlah karakter tiap pesan yang dikirim. Selain itu, media SMS juga menggunakan transaksi berbayar dengan pemotongan pulsa. Adapun media lainnya yang memberikan fitur lebih yaitu menggunakan media aplikasi *instant messaging*. Dalam penelitian ini memanfaatkan aplikasi *instant messaging* sebagai media peringatan pemberitahuan terhadap adanya intrusi dalam sistem.

2.4 *Instant Messaging* Telegram

Aplikasi *instant messaging* saat ini sangat populer di kalangan masyarakat. Tujuan utama aplikasi tersebut yaitu menyajikan fitur obrolan yang berjalan secara *real time* sehingga pesan langsung dapat terkirim dan diterima. Aplikasi *instant messaging* berjalan secara *online* atau dengan kata lain membutuhkan koneksi *Internet*. Saat ini terdapat banyak aplikasi *instant messaging* yang digunakan oleh masyarakat untuk mengobrol dengan individu maupun komunitas. Fitur yang disajikan aplikasi tersebut tidak hanya melalui *text based* saja, tetapi bisa juga untuk melakukan obrolan melalui suara, bertukar foto, audio, video hingga dokumen digital. Salah satu aplikasi yang memiliki fitur tersebut yaitu Telegram.

Telegram secara definisi menurut telegram.org (Vico, 2014) merupakan alternatif layanan aplikasi perpesanan untuk ponsel (*mobile*) maupun *dekstop* yang

berbasis *cloud* dengan tingkat keamanan tinggi serta kecepatan aksesnya. Aplikasi *intant messaging* tersebut tersedia untuk berbagai *device* seperti ponsel yang memiliki sistem operasi Android, iOS, Windows Phone, Ubuntu Touch. Tidak hanya dapat digunakan pada perangkat *mobile*, tetapi juga dapat berjalan pada sistem *desktop* seperti Windows, OS X, dan Linux. Meskipun aplikasi tersebut terlihat sederhana, tetapi gratis dan memiliki fitur yang lebih unggul dibandingkan dengan aplikasi *instant messaging* lainnya. Telegram diklaim sebagai aplikasi yang aman karena salah satunya memiliki fitur dimana menyediakan pilihan pesan *end to end* yang dienkripsi serta dapat hancur dengan sendirinya dalam jangka waktu tertentu.

2.4.1 Telegram *Bot*

Aplikasi *instant messaging* Telegram memiliki *Application Progamming Interface* (API) yang dapat digunakan oleh publik. Berbeda dengan *instant messaging* lainnya seperti WhatsApp dan LINE. Pada *instant messaging* WhatsApp tidak menyediakan API bagi publik, tetapi aplikasi LINE meyediakan API dengan versi *trial* atau terbatas. API yang disediakan oleh Telegram dapat digunakan oleh siapapun dan tanpa batas. Telegram juga memiliki *bot* API yang memungkinkan untuk dengan mudah membuat program yang menggunakan pesan Telegram sebagai antarmuka. API ini memungkinkan pengembang untuk menghubungkan *bot* pada sistem Telegram. Telegram *bot* merupakan cara khusus yang tidak memerlukan nomor telepon tambahan sebagai syarat khususnya. Akun *bot* tersebut berfungsi sebagai antarmuka untuk kode yang dapat dijalankan pada *server* pengembang. Bot tersebut dapat melakukan beberapa pekerjaan yaitu:

a. Mengintegrasikan dengan layanan lainnya

   Bot dapat mengirimkan komentar jarak jauh atau mengendalikan *smart home*. Selain itu, *bot* juga mampu mengirimkan pemberitahuan melalui Telegram ketika terjadi sesuatu di suatu tempat

b. Menciptakan alat khusus

   Bot mampu memberikan pemberitahuan maupun memberikan sebuah peringatan, ramalan cuaca, terjemahan, atau layanan lain.

c. Membangun *single player* ataupun *multiplayer game*

Keunggulan lainnya yaitu *bot* mampu memainkan permainan seperti catur.

d. Membangun layanan sosial

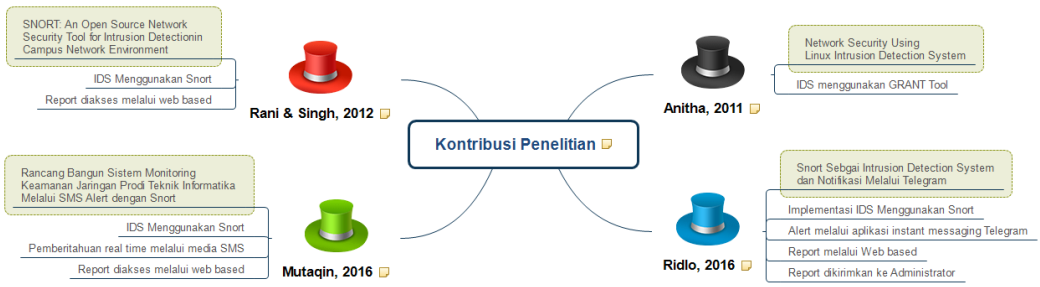
Sebuah *bot* dapat menghubungkan orang-orang untuk mencari mitra percakapan berdasarkan kepentingan bersama.

Dalam penggunaannya, pengembang tidak perlu repot untuk mengetahui protokol enkripsi Telegram karena hal tersebut akan ditangani oleh API Telegram. API Telegram berupa sebuah kode otentikasi yang disebut *token*. *Token* tersebut didapatkan ketika telah melakukan pendaftaran akun pada Telegram. Pada implementasinya, pengembang hanya memerlukan *token* sebagai syarat untuk menggunakan Telegram *bot.*

Pada Telegram *bot* API tersedia beberapa metode dalam pegiriman pesan yaitu *getMe*, *sendMessage*, *sendDocument*, *sendPhoto*, dan lain-lain ("All Method," n.d.). Setiap metode tersebut harus memiliki parameter *chat_id* yang mendefinisikan identitas target obrolan. Namun, terdapat perbedaan parameter pada setiap metode misalnya *sendMessage* wajib memiliki parameter *text* yang memiliki nilai berupa pesan yang akan dikirim. Sedangkan *sendDocument* harus memiliki parameter *document* yang berisi *file* yang akan dikirimkan. Penelitian ini memanfaatkan metode *sendMessage* untuk mengirimkan notifikasi singkat dan metode *sendDocument* untuk mengirimkan *attachment file* dalam format PDF.

2.5 Kontribusi Penelitian

Penelitian akan lebih mudah dipahami dengan adanya *Mind Map* yang berfungsi sebagai pola pikir yang digambarkan menjadi poin-poin penting. *Mind Map* yang dibuat mewakili atau menggambarkan tentang aspek-aspek yang terdapat dalam penelitian salah satunya penelitian mengenai keamanan jaringan. Adapun penelitian lain yang membahas tentang keamanan jaringan dengan menggunakan IDS yang dilakukan oleh beberapa peneliti yang dapat dilihat pada Gambar 2.3.



Gambar 2.3 Kontribusi Penelitian

Pada *Mind Map* kontribusi penelitian di atas digambarkan bagaimana penelitian IDS dibangun untuk keamanan jaringan. Kontribusi dari masing-masing penelitian yaitu:

a. **(Rani & Singh, 2012)** melakukan penelitian mengenai IDS dengan menggunakan Snort dimana laporan diakses melalui *web based*.
b. **(Anitha, 2011)** membangun IDS dengan menggunakan GRANT Tool.
c. **(Mutaqin, 2016)** mengimplementasikan IDS menggunakan Snort dan alert melalui media SMS.

Beberapa penelitian di atas terdapat perbedaan mengenai laporan yang disajikan. Adapun yang hanya menggunakan *log* aktifitas intrusi dan juga terdapat *log* yang dikirimkan ke administrator melalui media tertentu. Terdapat penelitian yang hanya berfokus pada deteksi intrusi dan tidak memberikan sebuah peringatan kepada administrator.

Penelitian lain yang dilakukan oleh Mutaqin (2016) memanfaatkan SMS sebagai media peringatan terhadap intrusi. Fitur yang disediakan hanya sebatas pemberitahuan singkat melalui SMS yang berbayar dan terbatas. Hal tersebut kurang efisien dan sistem administrator harus melakukan pengecekkan laporan pada *server*. Maka dari itu, penelitian ini memanfaatkan media notifikasi yang *real time*, efisien, dan mampu mengirimkan laporan dalam jangka waktu tertentu. Hal tersebut dapat dilakukan menggunakan aplikasi *instant messaging* Telegram yang memanfaatkan *bot* sebagai media komunikasi antara *server* dan

administrator. Telegram *bot* tidak hanya dapat melakukan komunikasi terhadap satu orang, tetapi dapat memberikan pemberitahuan melalui sebuah grup.