

**PROYEK AKHIR**

**GERRY ITALIANO WOWILING**

**NRP. 2110135027**

Dosen Pembimbing:

Ferry Astika Saputra, ST, M.Sc

NIP. 197708232001121002

Isbat Uzzin Nadhori, S.Kom, MT

NIP. 197405052003121002

**PROGRAM STUDI D4 LJ TEKNIK INFORMATIKA**

**DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2014**

# **ABSTRAK**

*Monitoring* jaringan internet menggunakan teknologi *Intrution Detection Sytem* (IDS) sudah banyak diterapkan oleh para pelaku *network security*. Teknologi IDS yang ada saat ini kebanyakan menggunakan sistem *Signature Based* dimana *vendor* tiap-tiap IDS yang menyediakan klasifikasi dari *signature* sebuah *events.*

**Kata-kunci**: *Monitoring, Internet, Intrution Detection System, Modularity*

# **ABSTRACT**

**Keywords**: *Monitoring, Internet, Intrution Detection System, Modularity*

# **DAFTAR ISI**

[ABSTRAK ii](#_Toc410312409)

[ABSTRACT iii](#_Toc410312410)

[DAFTAR ISI iv](#_Toc410312411)

[DAFTAR TABEL vii](#_Toc410312412)

[DAFTAR GAMBAR viii](#_Toc410312413)

[BAB I 1](#_Toc410312414)

[PENDAHULUAN 1](#_Toc410312415)

[**1.1.** **Latar Belakang Masalah** 1](#_Toc410312416)

[**1.2.** **Perumusan Masalah** 2](#_Toc410312417)

[**1.3.** **Tujuan** 2](#_Toc410312418)

[**1.4.** **Kontribusi** 3](#_Toc410312419)

[**1.5.** **Metodologi** 3](#_Toc410312420)

[**1.5.1.** **Studi Literatur** 4](#_Toc410312421)

[**1.5.2.** **Pengumpulan Data** 4](#_Toc410312422)

[**1.5.3.** **Perancangan Sistem** 4](#_Toc410312423)

[**1.5.4.** **Uji Coba Aplikasi** 4](#_Toc410312424)

[**1.5.5.** **Penarikan Kesimpulan** 4](#_Toc410312425)

[**1.5.6.** **Penyusunan Laporan** 5](#_Toc410312426)

[**1.6.** **Sistematika Penulisan** 5](#_Toc410312427)

[**Bab I**  **PENDAHULUAN** 5](#_Toc410312428)

[**Bab II**  **LANDASAN TEORI** 5](#_Toc410312429)

[**Bab III** **DESAIN APLIKASI** 5](#_Toc410312430)

[**Bab IV** **PENGUJIAN DAN ANALISA** 5](#_Toc410312431)

[**Bab V** **PENUTUP** 6](#_Toc410312432)

[**DAFTAR PUSTAKA** 6](#_Toc410312433)

[BAB II 7](#_Toc410312434)

[LANDASAN TEORI 7](#_Toc410312435)

[**2.1.** **Dasar Permasalahan** 7](#_Toc410312436)

[**2.2.** **Penelitian Terkait** 10](#_Toc410312437)

[**2.3.** **Keunikan** 11](#_Toc410312438)

[**2.4.** **Intrution Detection System** 14](#_Toc410312439)

[**2.4.1.** **NIDS (Network Based IDS) :** 14](#_Toc410312440)

[**2.5.** **SNORT** 16](#_Toc410312441)

[***2.6.*** ***Open Source* DBMS PostgreSQL** 18](#_Toc410312442)

[***2.7.*** ***Web Application*** 19](#_Toc410312443)

[**2.7.1.** **Framework** 20](#_Toc410312444)

[**2.7.2.** **CodeIgniter** 20](#_Toc410312445)

[***2.8.*** ***Network Forensic Challanges*** 22](#_Toc410312446)

[BAB III 24](#_Toc410312447)

[DESAIN APLIKASI 24](#_Toc410312448)

[**3.1.** **Deskripsi Umum Sistem** 24](#_Toc410312449)

[**3.2.** **Perancangan Sistem** 27](#_Toc410312450)

[**3.1.1.** **Rancangan Aplikasi** 27](#_Toc410312451)

[**3.1.2.** **Use Case Diagram** 30](#_Toc410312452)

[**3.1.3.** **Entity Relationship Diagram** 33](#_Toc410312453)

[**3.3.** **Kebutuhan Pendukung Pengembangan** 45](#_Toc410312454)

[**3.2.1.** **Kebutuhan Perangkat Keras** 45](#_Toc410312455)

[**3.2.2.** **Kebutuhan Perangkat Lunak** 45](#_Toc410312456)

[**3.4.** **Pengembangan Aplikasi** 49](#_Toc410312457)

[**3.3.1.** **Instalasi dan Konfigurasi Snort** 50](#_Toc410312458)

[**3.3.2.** **Membangun Database IDS Snort** 54](#_Toc410312459)

[**3.3.3.** **Monitoring Internet (Capture Packet)** 57](#_Toc410312460)

[**3.3.4.** **Membangun Aplikasi Menggunakan PHP dan CodeIgniter** 59](#_Toc410312461)

[BAB IV 71](#_Toc410312462)

[PENGUJIAN SISTEM DAN APLIKASI 71](#_Toc410312463)

[**4.1.** **Pengujian Terhadap IDS Snort** 71](#_Toc410312464)

[**4.1.1.** **Pengujian Konektifitas** 71](#_Toc410312465)

[**4.1.2.** **Pengujian Running Snort** 72](#_Toc410312466)

[**4.1.3.** **Pengujian Catpure Events** 74](#_Toc410312467)

[**4.2.** **Pengujian Database** 75](#_Toc410312468)

[**4.2.1.** **List Table pada Database** 76](#_Toc410312469)

[**4.2.2.** **Pengujian Query** 76](#_Toc410312470)

[**4.3.** **Pengujian Aplikasi Mata Garuda** 77](#_Toc410312471)

[**4.3.1.** **Pengujian Menu Dashboard** 78](#_Toc410312472)

[**4.3.2.** **Pengujian Menu Events** 81](#_Toc410312473)

[**4.3.3.** **Pengujian Menu Event Statistic** 82](#_Toc410312474)

[**4.3.4.** **Pengujian Menu Report** 84](#_Toc410312475)

[BAB V 85](#_Toc410312476)

[PENUTUP 85](#_Toc410312477)

[**5.1.** **Kesimpulan** 85](#_Toc410312478)

[**5.2.** **Saran** 85](#_Toc410312479)

[BIODATA PENULIS 88](#_Toc410312480)

# **DAFTAR TABEL**

[**Gambar 1Topologi NIDS Snort** 15](#_Toc410312481)

[**Gambar 2 Log IDS Snort** 17](#_Toc410312482)

[**Tabel 1 Tabel Limit PostgreSQL** 19](#_Toc410312483)

[**Gambar 3 Struktur Direktori CodeIgniter** 21](#_Toc410312484)

# **DAFTAR GAMBAR**

[**Gambar 1Topologi NIDS Snort** 15](#_Toc410312485)

[**Gambar 2 Log IDS Snort** 17](#_Toc410312486)

[**Tabel 1 Tabel Limit PostgreSQL** 19](#_Toc410312487)

[**Gambar 3 Struktur Direktori CodeIgniter** 21](#_Toc410312488)

# **BAB I**

# **PENDAHULUAN**

## **Latar Belakang Masalah**

*Monitoring* intrusi menjadi sebuah kajian penting dalam suatu jaringan. Penerapan sistem intrusi pada *monitoring* *network* banyak dilakukan karena *monitoring* intrusi akan memberikan sebuah *alerts* atau peringatan kepada administrator terhadap serangan yang datang ke sebuah jaringan. Sistem *monitoring* intrusi tersebut bekerja sesuai dengan *rules* yang ditetapkan administrator. Tool atau sistem *monitoring* yang paling sering digunakan adalah IDS Snort, IDS Snort bekerja dengan cara meng-*capture* paket pada jaringan, mengambil informasi yang dibutuhkan lalu mencocokan dengan *rule* yang telah ditetapkan, jika terjadi pelanggaran terhadap *rules* maka sistem IDS Snort akan memberikan *alerts* kepada administrator bahwa terdapat paket yang mencurigakan dan hasilnya dapat dianalisa oleh administrator*.*

*Alerts* yang diberikan oleh IDS Snort masih berupa *text based,* hal tersebut terkadang menyulitkan administrator untuk melakukan analisa terlebih jika datanya sangat banyak*,* sehingga diperlukan adanya aplikasi tambahan atau *interface* yang bisa menyediakan tampilan yang sesuai dengan yang diingankan pihak administrator untuk memudahkan analisa hasil *monitoring*. Beberapa *tools* atau aplikasi yang menyediakan *interface* untuk menampilkan *alerts* IDS Snort adalah BASE (Basic Analysis Security Engine), SGUIL (Snort GUI for Lamers), dan Snorby. Aplikasi-aplikasi tersebut dibangun dari beberapa bahasa pemrograman yaitu Perl, PHP, ataupun Ruby, namun memiliki kendala dalam sisi pengembangan dimana aplikasi tersebut belum menerapkan sistem modular sehingga jika ada pengembangan lebih lanjut harus mengubah atau mengerjakan keseluruhan aplikasi tanpa bisa dikerjakan bagian per bagian.

Selama ini belum ada aplikasi yang memberikan kemudahan pengembangan *interface* IDS Snort dalam bentuk modular, sehingga diperlukan suatu aplikasi *interface* (dalam bentuk web) berbasis IDS Snort yang mampu menjawab masalah tersebut.

## **Perumusan Masalah**

Pentingnya menyajikan data hasil *monitoring* kepada administrator jaringan dalam bentuk yang mudah dibaca, menjadi sebuah permasalahan yang coba dikembangkan oleh penulis dalam bentuk perancangan aplikasi *interface* sehingga memudahkan administrator untuk melakukan analisa. Masalah tersebut dirumuskan menjadi beberapa poin-poin seperti berikut:

1. Bagaimana membuat atau merancang sebuah aplikasi *interface* berbasis IDS Snort dan Postgre untuk melakukan monitoring jaringan internet.
2. Bagaimana membuat atau mendesain sebuah *web application modular* untuk memudahkan pengembangan aplikasi.

Bagaimana mengembangkan suatu sistem yang sesuai dengan kebutuhan monitoring network.

## **Tujuan**

Adapun tujuan dari proyek akhir ini antara lain sebagai berikut:

1. Merancang dan membangun suatu aplikasi *interface* untuk IDS berbasis Snort dan Postgre.
2. Merancang aplikasi berbasis web modular yang bisa dikembangkan bagian per bagian tanpa mengganggu kinerja keseluruhan aplikasi.
3. Mengembangkan suatu sistem yang sesuai dengan kebutuhan monitoring *networking.*

## **Kontribusi**

Melalui rancang bangun aplikasi Mata Garuda, diharapkan mampu untuk menyediakan dan memfasilitasi sebuah *web application* yang akan memberikan hasil analisa serangan IDS Snort kedalam bentuk yang lebih *user friendly* bagi administrator*.* Aplikasi yang dikembangkan nantinya juga dapat fleksibel terhadap pengembangan-pengembangan lebih lanjut, dalam artian aplikasi *interface* (dalam bentuk web) yang dibangun harus mampu menyediakan suatu cara agar pengembangan selanjutnya tidak mengalami kendala dengan batasan-batasan yang ada pada *web application* yang dibangun. Dengan demikian jika terjadi pengembangan-pengembangan selanjutnya, tidak perlu memulai dari awal namun dapat langsung fokus terhadap poin-poin penting yang akan dikembangkan.

Melihat pentingnya suatu sistem monitoring untuk mengatasi intrusi *traffic* internet, rancang bangun aplikasi *interface* Mata Garuda berbasis IDS Snort sangat diperlukan untuk memberikan kemudahan kepada pengguna (terutama administrator jaringan) membaca dan mengolah informasi yang disediakan oleh Mata Garuda sehingga dampak buruk yang muncul dalam penggunaan internet dapat diminimalisir*.*

## **Metodologi**

Dalam membangun dan merancang sebuah aplikasi erat kaitannya dengan *Software Development Life Cycle* (SDLC) yaitu sebuah struktur yang ada pada pengembangan sebuah *sofware,* mengembangkan atau mengubah sistem dari *software* tersebut dengan menggunakan model dan metodologi yang digunakan pada pengembangan sistem perangkat lunak sebelumnya.

Salah satu model yang sering digunakan pada SDLC yaitu *waterfall* dimana pada model ini menggunakan alur perangkat lunak secara beruntun. Begitu juga pada pengembangan Project Mata Garuda ini menggunakan beberapa tahapan yang dilakukan secara berurutan mulai dari studi literatur, pengumpulan data, perancangan sistem, uji coba aplikasi, penarikan kesimpulan, hingga penyusunan laporan.

### **Studi Literatur**

Mengumpulkan dan mempelajari referensi mengenai *monitoring* jaringan internet menggunakan berbagai macan *Intrution Detection System* (IDS) yang bersifat *open source*, mempelajari *interface* yang telah ada sebagai salah satu kemudahan untuk analisa data.

### **Pengumpulan Data**

Mengumpulkan data-data yang terkait dengan *monitoring* jaringan berupa hasil *log* yang dihasilkan oleh sistem IDS *open source*. Data-data tersebut dapat diperoleh pada folder penyimpanan *log* ataupun pada *database* IDS *open source.* Khusus untuk penyimpanan *database,* memerlukan konfigurasi tersendiri pada proses instalasi IDS *open source*.

### **Perancangan Sistem**

Pada tahap ini melakukan perancangan system implementasi yang akan merancang sebuah aplikasi antar muka IDS *open source* menggunakan bahasa pemrograman PHP dan memanfaatkan *Database Management System* (DBMS) PostgreSQL. Aplikasi antar muka yang dirancang bersifat *modular framework,* dalam artian aplikasi dapat dikembangkan bagian per bagian (modul per modul) secara terpisah tanpa harus mengganggu keseluruhan sistem.

### **Uji Coba Aplikasi**

Setelah dilakukan perancangan dan implementasi, aplikasi hanya akan mengambil data yang ada pada *database* IDS, data yang ada pada folder log sistem hanya digunakan jika memang dibutuhkan oleh administrator jaringan, aplikasi hanya memberikan informasi letak log tersebut berada. Tahap pengujian aplikasi sangat diperlukan untuk mengetahui apakah aplikasi yang telah dirancang dan dibangun sesuai dengan kebutuhan awal pengembangan atau tidak.

Data-data yang telah tersimpan di dalam *database* IDS akan ditampilkan kedalam bentuk yang lebih variatif dan mudah dibaca oleh administrator untuk memudahkan proses analisa.

### **Penarikan Kesimpulan**

Berdasarkan hasil uji coba, diambil kesimpulan bahwa aplikasi yang dibangun dapat memberikan kemudahan bagi seorang administrator untuk dapat melakukan analisa data hasil *monitoring* jaringan internet. Data *monitoring* yang sebelumnya sulit untuk dibaca dan dilakukan analisa menjadi lebih mudah karena adanya antar muka aplikasi Mata Garuda yang dikembangkan.

### **Penyusunan Laporan**

Membuat dokumentasi dari semua tahapan proses diatas berupa laporan yang berisi tentang teori dan hasil proyek akhir ini.

## **Sistematika Penulisan**

### **Bab I PENDAHULUAN**

Bab ini berisi penjelasan singkat mengenai latar belakang masalah, perumusan masalah, tujuan, metodologi, dan sistematika pembahasan.

### **Bab II LANDASAN TEORI**

Bab ini berisi penjelasan mengenai *monitoring* jaringan internet, IDS berbasis *open source,* DBMS sebagai salah satu media penyimpanan, *web application* yang memberikan kemudahan akses aplikasi beserta *framework* dan bahasa pemrograman yang digunakan yaitu PHP, serta penjelasan materi mengenai bentuk pengembangan *modular* yang dapat diterapkan pada *web application* berbasis *framework* PHP.

### **Bab III DESAIN APLIKASI**

Bab ini berisi penjelasan mengenai cara mengkonfigurasi dan mengimplementasikan IDS berbasis *open source* pada jaringan internet yang akan dipantau, rancang bangun aplikasi antar muka hasil *monitoring* IDS menggunakan *framework* PHP dan pemanfaatan DBMS sebagai media penyimpanan data-data yang telah dihasilkan tersebut.

### **Bab IV PENGUJIAN DAN ANALISA**

Pada bab ini dibahas tentang pengujian aplikasi antar muka IDS berbasis *open source* yang menyajikan data-data berupa *log* dari *database* yang dihasilkan oleh sistem IDS.

### **Bab V PENUTUP**

Pada bab ini dibahas mengenai kesimpulan dari system terkait dengan tujuan dan permasalahan yang ada, serta saran untuk pengembangan system di masa mendatang.

### **DAFTAR PUSTAKA**

Berisi tentang referensi-referensi yang telah digunakan sebagai landasan selama pembuata Proyek Akhir ini.

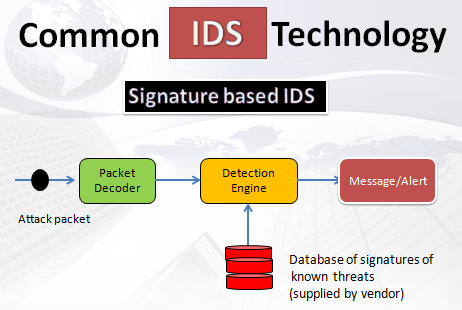
# **BAB II**

# **LANDASAN TEORI**

## **Dasar Permasalahan**

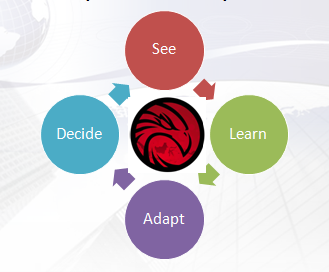
Internet merupakan salah satu media komunikasi yang paling banyak digunakan oleh hampir semua teknologi terutama perangkat komputer, jaringan internet menghubungkan banyak perangkat komputer untuk saling berkomunikasi satu dengan lainnya sehingga lalu lintas data yang ada pada jaringan internet sangat padat. Komunikasi data yang sangat banyak tersebut perlu dilakukan pemantauan atau *monitoring* untuk menghindari tindak kejahatan yang mungkin terjadi pada penggunaan teknologi internet. Salah satu *tool* atau sistem yang paling sering digunakan untuk melakukan *monitoring* jaringan internet adalah *Intrution Detection System* (IDS). IDS tersebut melakukan *monitoring* jaringan internet berdasarkan *rule* atau atauran-aturan yang telah ditetapkan administrator pada saat konfigurasi begitu juga dengan alamat IP spesifik dari jaringan internet yang dipantau, hasil *monitoring* berupa log-log informasi mengenai paket data yang berhasil dipantau, log tersebut dapat disimpan pada folder sistem atau dapat juga disimpan pada *database.* Hasil log IDS tersebut yang menjadi bahan kajian bagi para administrator jaringan untuk melakukan analisa lebih lanjut. Namun pada sistem IDS ini masih dijumpai permasalahan dan kendala yaitu data yang dihasilkan IDS masih dalam bentuk *text-based* yang tentunya masih menyulitkan administrator untuk menganalisa dan mendapatkan informasi dari data tersebut, sehingga perlu adanya sebuah aplikasi antar muka yang mampu menjembatani antara sistem IDS dan administrator. Beberapa aplikasi antar muka yang mampu menyajikan data ­*text-based* tersebut kedalam bentuk tampilan yang lebih variatif adalah Snorby yang dibangun menggunakan bahasa pemrograman Ruby dan Perl, ACID/BASE yang dibangun menggunakan bahasa pemrograman PHP namun tidak memanfaatkan *framework*. Untuk memenuhi kebutuhan *monitoring* perlu dibangun dan dikembangkan lagi aplikasi antar muka yang sesuai dengan kebutuhan administrator sehingga kegiatan *monitoring* dan analisa data pada jaringan internet menjadi sebuah pekerjaan yang menyenangkan.

Permasalahan lain yang menjadi landasan pengembangan Proyek Akhir ini ialah teknologi yang dimiliki oleh beberapa *Intrution Detection* yang ada saat ini (*signature based*) dirasa masih belum bisa untuk memenuhi kebutuhan administrator akan sebuah teknologi IDS yang muktahir, namun juga tidak bisa disingkirkan fakta bahwa teknologi IDS yang ada saat sudah cukup membantu administrator untuk melakukan analisa *traffic* jaringan internet.



Dari Gambar diatas dapat dijelaskan bahwa IDS yang ada saat ini mengandalkan *signatures* dari beberapa ancaman yang diketahui hanya berbasis pada informasi yang dikeluarkan oleh *vendor,* sementara kebutuhan akan analisa jaringan (termasuk didalamnya *signatures*) bisa berbeda-beda tergantung regional.

Perlu adanya sebuah teknologi baru yang mampu memenuhi kebutuhan administrator jaringan untuk menganalisa jaringan dan melakukan *forensic security network.*



Pada gambar diatas dapat dijelaskan beberapa tahapan dari teknologi yang dibutuhkan:

1. **See** : Teknologi IDS yang diharapkan dapat melihat semua *packet* data yang melalui jaringan, bukan hanya pada saat serangan namun juga pada saat normal sehingga data yang tersimpan mencakup keseluruhan informasi *packet* data.
2. **Learn** : Menambah pemahaman terhadap karakteristik *traffic* jaringan lokal dengan menerapkan teknologi *machine learning* untuk memahami *intelligence traffic* jaringan.
3. **Adapt** : Mengembangkan suatu *tools* jaringan yang bersifat *modular* untuk mempermudah analisa
4. **Decide** : Mengurangi *false alarm* dari *traffic* jaringan dengan memanfaatkan teknologi *intelligence.*

Dari beberapa kebutuhan yang dipaparkan diatas, pada Proyek Akhir ini penulis hanya berfokus pada “*Adapt*”, yaitu membangun *core sytem* aplikasi yang bersifat *modular* dengan berbasis IDS *open source.*

## **Penelitian Terkait**

Beberapa penelitian atau pengembangan aplikasi yang telah dilakukan oleh berbagai tim pengembang lain diantaranya adalah :

1. **Emmanuel S Pilli, R.C. Joshi, Rajdeep Niyogi, Network Forensic Frameworks: Survey and Research Challenges[**1**]**

Pada jurnal tersebut, Pilli, Joshi, dan Niyogi membahasa tentang fungsi dan kegunaan dari berbagai macam *Network Forensic Analysis Tools* (NFATs) dan *Network Security Monitoring Tools* (NSM) setelah sebelumnya melakukan survei terkait untuk masing *tools* yang mereka bahas. Selain membahas fungsi dan kegunaan dari tiap-tiap NFAT dan NSM yang ada sekarang ini, mereka juga mengajukan suatu mekanisme *generic process model* dari berbagai macam *tools networking* untuk dapat melakukan analisa forensik jaringan. Latar belakang jurnal mereka tersebut didasari oleh kenyataan bahwa *network forensics* telah dan sedang dalam penelitian untuk beberapa dekade namun masih dianggap sebagai ilmu pengetahuan yang belum memadai (*young science*) dan masih banyak permasalahan pada bidang ini yang belum jelas dan ambigu. Tujuan mereka mereka mengemukakan jurnal ini ialah untuk mempermudah para praktisi dan peneliti keamanan jaringan dalam mengeksplorasi berbagai isu jaringan yang ada.

1. **Roman Danyliw, Analysis Console for Intrusion Databases (ACID)**[2]**.**

ACID merupakan sebuah sistem/aplikasi analisa berbasis pemrograman PHP untuk melakukan pencarian dan mengolah data pada *database* yang berkaitan dengan permasalahan keamanan jaringan, dimana data tersebut dihasilkan oleh berbagai macam sistem dan perangkat keamanan jaringan seperti IDS ataupun *firewall*. ACID memiliki beberapa fitur unggulan termasuk diantaranya yaitu: Antar Muka untuk pencarian dan *Query-Builder, Packet Viewer, Alert Management,* hingga menampilkan grafik dan statistik berdasarkan masukkan waktu dan sebagainya. ACID memiliki kemampuan untuk menganalisa berbagai macam kejadian yang berkaitan dengan keamanan jaringan yang telah terlebih dahulu disimpan kedalam *database* oleh SNORT ataupun longSnorter sebagai sistem IDS yang digunakan.

1. **Kevin Johnson, Basic Analysis and Security Engine (BASE)**[3].

BASE merupakan aplikasi antar muka IDS perkembangan lebih lanjut dari ACID sehingga basis programnya tetap menggunakan basis program ACID. Sejak ACID mulai ditinggalkan pada tahun 2003, Kevin Johnson meneruskan pengembangannya namun memilih menggunakan nama baru (BASE) dan menambahkan beberapa fitur-fitur baru ke dalam BASE sehingga lebih mudah digunakan serta merupakan antar muka aplikasi yang dapat berjalan dengan baik. Namun sama halnya seperti ACID, BASE juga tidak lagi dikembangkan walaupun memegang predikat sebagai aplikasi antar muka SNORT yang paling populer dan sudah lebih dari 215.000 pengguna yang mengunduhnya. Pada masa pengembangannya BASE memiliki saluran IRC (*Internet Relay Chat*) yang memudahkan pengguna untuk berkomunikasi secara langsung dengan tim pengembang dari BASE.

1. **Snorby[**4**]**

Snorby merupakan aplikasi antar muka berbasis web yang menyajikan data-data *monitoring* IDS kepada administrator. Snorby bersifat ­*front-end* aplikasi yaitu hanya menyajikan data yang dihasilkan oleh IDS saja, dengan demikian segala konfigurasi dan pengaturan yang ada pada IDS ataupun perangkat *monitoring* lainnya tidak bisa dilakukan pada antar muka Snorby. Snorby dibangun dengan menggunakan bahasa pemrograman Ruby dan Rails dan banyak menggunakan efek atau fungsi “Web 2.0” untuk memberikan pengguna berbagai macam kemudahan dan pilihan *tool.*

## **Keunikan**

Dari permasalahan dan penelitian terkait yang telah dipaparkan sebelumnya, Proyek Akhir Rancang Bangun Aplikasi Mata Garuda ini memiliki beberapa keunikan yang belum ada pada penelitian sebelumnya, sehingga layak untuk dibangun dan dikembangkan. Berikut ini merupakan keunikan yang dimiliki oleh Aplikasi Mata Garuda:

1. *Framework PHP*

Aplikasi Mata Garuda yang merupakan aplikasi antar muka IDS berbasis *open source* dibangun dan dikembangkan menggunakan bahasa pemrograman PHP, namun tidak seperti ACID/BASE yang juga dikembangkan menggunakan bahasa pemrograman PHP pada pengerjaan Proyek Akhir Rancang Bangun Aplikasi Mata Garuda ini menerapkan sistem *framework,* yaitu sebuah rangkaian fungsi-fungsi yang telah disediakan oleh beberapa pengembang *framework* PHP untuk memudahkan pembangunan aplikasi berbasis pemrograman PHP semakin mudah.

1. *Modularity*

Pada proyek akhir ini menggunakan *framework* PHP dalam pengembangannya sehingga memungkinkan untuk diterapkan suatu metode yang dinamai *modularity. Modularity* merupakan suatu cara dalam pengembangan aplikasi berbasis PHP yang melakukan pembagian atau pemisahan tiap-tiap fungsi untuk dikerjakan secara terpisah dan mandiri namun tetap menjadi satu bagian sistem secara keseluruhan. Dalam artian, tiap-tia fungsi yang ada pada Aplikasi Mata Garuda ini dapat dikembangkan secara terpisah tanpa harus mengubah kinerja dan fungsi sistem secara keseluruhan.

Keunggulan dari penggunaan metode *modularity* ialah:

1. Fungsi dapat dikerjakan tanpa harus menunggu fungsi lainnya (fungsi tidak saling terkait).
2. Pengembangannya dapat dilakukan dengan cepat karena pembagian fungsi yang memungkinkan fungsi yang berbeda dikerjakan secara bersama.
3. Fungsi baru yang akan diterapkan pada aplikasi dapat dikerjakan secara terpisah diluar *environment* aplikasi utama Mata Garuda.
4. Jika terdapat fungsi yang dirasa kurang tepat pada aplikasi yang sedang berjalan, fungsi tersebut dapat dihilangkan tanpa harus menghentikan kinerja dari aplikasi.
5. *Open Source DBMS*

Penggunaan *Database Management System* merupakan hal yang umum dan terbilang utama pada pembangunan dan pengembangan sebuah aplikasi terutama aplikasi berbasis *web application.* Pada proyek akhir ini penulis menggunakan DBMS yang bersifat *open source* namun tetap memiliki berbagai macam keunggulan yang dimiliki oleh DBMS berbayar dengan tujuan bahwa aplikasi yang dibangun dapat menghemat dan mengefesiensi biaya.

*Database* yang digunakan dirancang sedemikian mungkin agar dapat menerima relasi atau perubahan yang mungkin dilakukan kedepannya dan tetap tidak mengubah struktur inti dari *database* dan aplikasi.

1. *Sensor*

Aplikasi Mata Garuda ini dirancang untuk dapat menampung dan menampilkan data monitoring dari beberapa *sensor,* sehingga memudahkan administrator jaringan untuk melakukan analisa dan perbandingan. Keuntungan lainnya, hanya dibutuhkan satu aplikasi antar muka untuk banyak *sensor* tanpa harus melakukan instalasi satu-persatu aplikasi antar muka pada tiap *sensor.* Aplikasi Mata Garuda nantinya juga akan menampilkan keterangan mengenai status *sensor.*

## **Intrution Detection System**

*Intrusion Detection System* (IDS) adalah sebuah perangkat komputer (baik *software* ataupun *hardware*) yang melakukan monitoring terhadap jaringan ataupun aktifitas pada sistem khususnya *traffic internet,* dan akan menghasilkan sebuah laporan atau peringatan kepada seorang administrator jaringan jika mendeteksi adanya sebuah tindak kejahatan ataupun aktifitas-aktifitas yang melanggar *policy* pada jaringan komputer. Sering kali IDS juga merespon terhadap trafik yang tidak normal/ *anomali* melalui aksi pemblokiran seorang user atau alamat IP (*Internet* *Protocol*) sumber dari usaha pengaksesan jaringan.

Ada beberapa metode dan pendekatan yang digunakan IDS untuk melakukan deteksi terhadap tindak kejahatan yang terjadi pada jaringan komputer, sistemyang biasanya digunakan yaitu *Network Based IDS (NIDS)*, *Protocol Based IDS (PIDS), Application Protocol Based IDS (APIDS),* dan juga *Host Based IDS (HIDS).* Pada pembahasan ini, yang akan dibahas adalah *Network base IDS*.

### **NIDS (Network Based IDS) :**

Sistem yang bekerja dengan cara menganalisis semua lalu lintas yang mengalir pada sebuah jaringan dan akan mencari apakah ada kemungkinan serangan atau intrusi di dalam lalu lintas tersebut, pada umumnya NIDS diletakan pada server yang berada di pintu masuk sebuah jaringan. NIDS memiliki sensor yang ditempatkan ke beberapa titik strategis di dalam jaringan untuk melakukan monitoring *traffic* dari dan menuju semua perangkat dalam jaringan. Melakukan analisis untuk melewatkan paket pada keseluruhan *subnet* , semua bentuk paket dianalisa dan melakukan *matching* paket terhadap *library* dari serangan-serangan yang telah diketahui

Fungsi Utama dari NIDS:

1. Mendeteksi Serangan

Sebagian sistem mendeteksi adanya ancaman *security* dan serangan, dan ketika hal tersebut terjadi sistem memberikan hasil monitoring jaringan secara *real-time.*

1. Memberikan Informasi

Sistem yang mendeteksi adanya serangan, akan memberikan informasi terkait dengan serangan tersebut.

1. Mengambil Langkah yang Sesuai

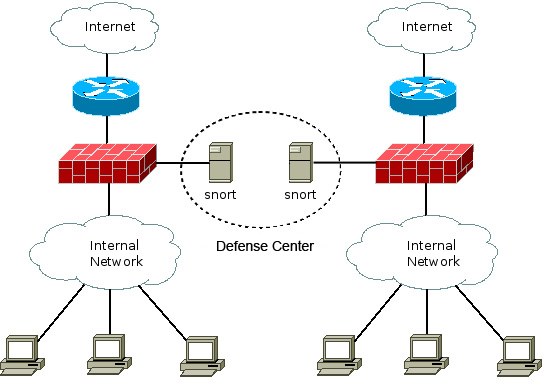
Ketika sebuah serangan berhasil dideteksi oleh sistem, *active system* lainnya juga akan melakukan perhitungan untuk menangkal serangan tersebut.

1. Penyimpanan

NIDS menyimpan events atau serangan-serangan yang terjadi untuk memudahkan analisa kedepannya.

**Topologi NIDS:**

Secara umum NIDS memiliki topologi jaringan seperti berikut:

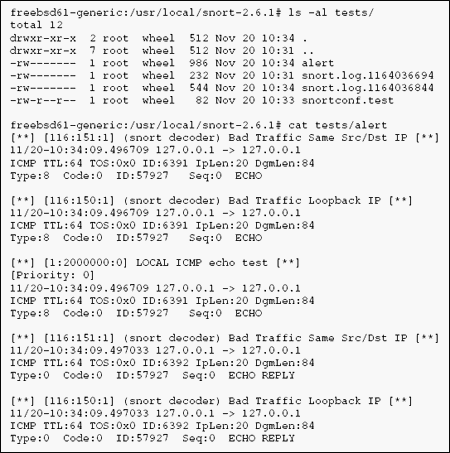
****

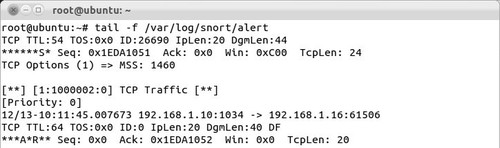
**Gambar 1Topologi NIDS Snort**

Pada Gambar 1 diatas terdapat 2 sensor IDS yaitu Snort yang ditempatkan di masing-masing network dan terhubung ke internet. Ketika ada serangan menuju suatu jaringan, IDS akan melakukan matching paket dengan rules yang telah ditentukan sebelumnya, jika paket tersebut dianggap sebagai intrusi maka IDS akan memberikan alerts kepada administrator dan menyimpannya ke dalam database atau log.

## **SNORT**

SNORT adalah salah satu contoh *software* dari NIDS yang melakukan *matching* paket terhadap *rule,* dan menentukan apakah sebuah paket atau serangan yang masuk ke dalam jaringan merupakan sebuah intrusi atau bukan. SNORT mampu melakukan analisis *traffic* dan *packet logging* secara *real-time* pada jaringan, mampu melakukan analisis protokol, *matching content,* dan juga dapat digunakan untuk mendeteksi berbagai jenis serangan dan pemeriksaan seperti halnya *buffer overflows, stealth port scan, CGI attacks, SMB Probes, OS Fingerprinting attemps,* dan masih banyak lagi terkait dengan percobaan penyerangan pada jaringan. SNORT akan memberikan *alerts* kepada administrator jika ada paket yang melalui jaringan dan dianggap sebagai sebuah serangan atau intrusi, berikut ini merupakan contoh *alerts* dari SNORT:





**Gambar 2 Log IDS Snort**

Pada Gambar 2 diatas merupakan contoh dari *alerts* SNORT yang memberikan informasi berupa kategori *traffic* (Bad Traffic), *source* IP Address dan Port, *destination* IP Address dan Port, Protokol, IpLen, dan informasi lainnya. Informasi tersebut dapat disimpan kedalam *database* dan dapat digunakan untuk kebutuhan analisa lebih lanjut

## ***Open Source* DBMS PostgreSQL**

PostgreSQL atau sering disebut sebagai Postgre merupakan object relational database management system (ORDBMS) yang terfokus kepada ekstensibiliti atau perluasan yang memungkinkan untuk menyimpan data dalam jumlah yang sangat besar. Postgre sebagai salah satu database server memiliki fungsi utama sebagai penyimpanan data, melakukan penyimpanan data dengan tingkat keamanan yang tinggi, dan dapat digunakan sewaktu-waktu oleh database administrator ataupun aplikasi lain yang terintegrasi dengan Postgre, baik pada localhost komputer ataupun pada host-host lain pada jaringan komputer yang terhubung (termasuk jaringan internet). Postgre dapat menangani beban kerja berkisar pada aplikasi komputer berskala kecil hingga aplikasi internet berskala besar dimana banyak pengguna yang terhubung pada waktu yang bersamaan. Fungsi lainnya yaitu dapat menduplikasikan database itu sendiri untuk alasan keamanaan dan pertumbuhan data.

PostgreSQL merupakan *database server multi-platform,* dengan demikian Postgre dapat berjalan pada hampir semua sistem operasi termasuk Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, MacOS X, Solaris, Tru64) dan tentunya Windows. Menggunakan format SQL:2011 standar yang tidak jauh beda dengan kebanyakan DBMS yang menangani data pada *server* ataupun *localhost,* termasuk didalamnya DDL *statements.* Postgre menggunakan hampir semua SQL:2008 standar *data type* yaitu Integer, Numeric, Boolean, Char, Varchar, Date, Interval, dan Timestamp, termasuk juga didalamnya data berupa *binary large objects* (gambar, suara, dan video). Kaitannya dengan *native programming,* Postgre memiliki antar muka yang dapat digunakan oleh C/C++, Java, .Net, Perl, Phtyon, Ruby, Tcl, dan ODBC.

Dibawah ini merupakan tabel *limit* dari PostgreSQL secara umum:

**Tabel 1 Tabel Limit PostgreSQL**

|  |  |
| --- | --- |
| **Limit** | **Value** |
| Maximum Database Size | Unlimited |
| Maximum Table Size | 32 TB |
| Maximum Row Size | 1.6 TB |
| Maximum Field Size | 1 GB |
| Maximum Rows per Table | Unlimited |
| Maximum Columns per Table | 250-1600 |
| Maximum Indexes per Table | Unlimited |

## ***Web Application***

*Web Application* adalah sebuah aplikasi atau software berbasis web yang biasanya dapat digunakan ketika sebuah host terhubung ke jaringan internet ataupun local network, menggunakan HTTP sebagai protokol utama untuk komunikasi dan pertukaran data serta memanfaatkan *web browser* untuk melakukan rendering aplikasi sehingga data dapat tersampaikan pada sisi *client* atau pengguna. Web Application terkadang bisa berupa client-based dalam artian bagian kecil dari sebuah program diunduh ke desktop pengguna, namun tetap proses utamanya terjadi pada jaringan internet di server aplikasi. *Web Application* biasanya dibangun menggunakan *programming language* yang mendukung seperti JavaScript, HTML, CSS, PHP, Perl, dan bahkan Java.

Web Application banyak digunakan karena kemudahan penggunaannya, dan juga dapat berjalan hampir di semua *platform,* dimana yang hanya dibutuhkan pada masing-masing *platform* ataupun sistem operasi hanyalah sebuah *browser* yang mampu menterjemahkan data pada sisi server sehingga dapat disajikan dalam bentuk informasi pada sisi *client.* Adanya *web application* juga memudahkan pada sisi pengembangan dan intergrasi dengan DBMS yang mendukung seperti PostgreSQL, MySQL, Oracle, dan database *enterprise* lainnya.

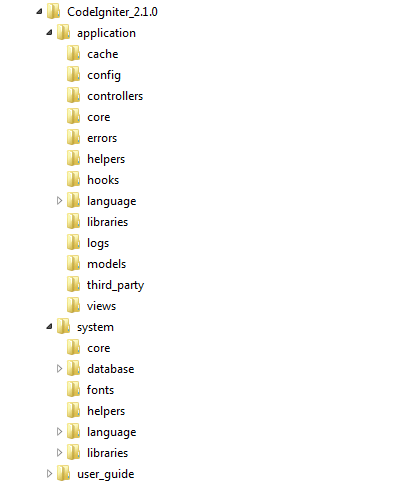
Untuk semakin memudahkan pengembangan, biasanya *web application* yang akan dibangun dibentuk menjadi sebuah rangka karja atau *framework* dimana *framework* ini dapat menerapkan sistem *modular* sehingga pengembangan dapat dilakukan bagian per bagian tanpa harus mengganggu kinerja aplikasi secara keseluruhan.

### **Framework**

Banyak sekali developer php mengembangkan dan menyempurnakan framework untuk memudahkan dalam membangun sebuah aplikasi berbasis web, diantaranya yang paling terkenal adalah CodeIgniter (CI) dan YesItIs (Yii) framework. Framework merupakan kumpulan dari berbagai fungsi dan class php yang menerapkan MVC (Model View Controler) pada pembangunannya, dengan menerapkan MVC pengembang dapat membuat design tampilan, design database, dan controller aplikasi secara terpisah, dengan cara ini diharapkan pembuatan aplikasi berbasis web dapat mengefesiensi waktu lebih baik, dan untuk pengembangan kedepannya juga dapat lebih mudah karena aplikasi yang dikembangkan langsung dapat terfokus pada modul atau bagian yang akan dikembangkan saja. Dengan adanya konsep modular pengembangan aplikasi akan lebih cepat diselesaikan karena pengerjaannya dapat berdiri sendiri sebelum nantinya digabungkan dengan aplikasi utama.

### **CodeIgniter**

Codeigniter merupakan suatu *framework* PHP yang sangat *powerfull* dengan sedikit *footprint,* dibangun untuk para pengembang yang membutuhkan peralatan sederhana namun elegan untuk menciptakan *web application* yang memiliki banyak fungsi. Codeigniter berbasis pola Model-View-Controller dalam pengembangannya, Controller digunakan untuk menangani semua fungsi pada suatu halaman web dan View digunakan untuk menyajikan data yang diolah oleh Controller. Kedua file tersebut merupakan syarat dasar agar web dapat digunakan dengan baik sementara Model bersifat opsional karena model menghubungkan database dengan web. Codeigniter banyak digunakan karena kemudahan fungsi yang diberikan serta banyaknya *library* yang bisa digunakan oleh pengembang aplikasi berbasis *web*



**Gambar 3 Struktur Direktori CodeIgniter**

Pada Gambar 3 diatas merupakan struktur direktori dari framework Codeigniter dimana direktori tersebut ditempatkan pada direktori hosting sebuah sistem operasi (htdocs atau www) agar dapat diakses melalui web browser.

## ***Network Forensic Challanges***

Pada subbab 2.2 (Penelitian Terkait, 1) telah dijelaskan tentang fungsi dan kegunaan dari *Network Forensic Analysis Tools* (NFTA) dan *Networks Security Monitoring* (NSM). Jurnal yang dibahas oleh Pilli, Joshi, Nayogi juga bertujuan untuk menjawab tantangan yang muncul dalam bidang *forensic* dan penelitian keamanan jaringan seperti : *Collection and Detection, Data Fusion and Examination, Analysis, Investigation,* dan *Incedent Response.*

Pada Proyek Akhir ini penulis berfokus hanya pada 2 tantangan yang diberikan yaitu *Collection and Detection* dan *Analysis.* Dua tantangan tersebut akan coba dipecahkan dengan membangun aplikasi antar muka IDS berbasis *open source,* Mata Garuda.

1. *Collection and Detection*

Tahap pertama dalam melakukan analisa *forensic* jaringan melibatkan kumpulan dari berbagai macam *traces* dan deteksi serangan pada jaringan. *Traces* yang dimaksud bisa berupa log dari IDS dan *firewall,* log yang dihasilkan oleh berbagai macam aplikasi dan *service* jaringan, hingga *packet* yang berhasil ditangkap oleh *sniffers.* Pada proyek akhir ini penulis berfokus pada pengumpulan data menggunakan IDS berbasis *open source* yaitu Snort, dimana Snort inilah yang nantinya melakukan *monitoring* jaringan, deteksi keamanan, dan menyimpan *packet-packet* yang dianggap berbahaya bagi jaringan.

1. *Analysis*

Tahap kritikal selanjutnya pada keseluruhan proses *forensic* jaringan yaitu untuk melakukan analisa data dan berakhir pada kesimpulan untuk menentukan keamanan sebuah jaringan internet. *Classification and Clustering* perlu diterapkan pada semua data yang berhasil ditangkap oleh IDS, tujuannya ialah untuk mempermudah melakukan analisa sesuai dengan kategori dan kebutuhan keamanan jaringan. Pada proyek akhir ini, proses klasifikasi dan klustering akan coba dipecahkan dengan membangun fungsi yang menampilkan keseluruhan *events,* statistik *events,* dan juga kebutuhan data transaksi (*report*).

# **BAB III**

# **DESAIN APLIKASI**

Pada bab ini, dibahas mengenai desain dan rancangan dalam pembangunan aplikasi Mata Garuda antar muka IDS berbasis *open source.* Desain dan perencanaan aplikasi ditujukan agar dalam pembangunan dan pengembangan aplikasi dapat berjalan sesuai dengan struktur yang telah didefenisikan sebelumnya.

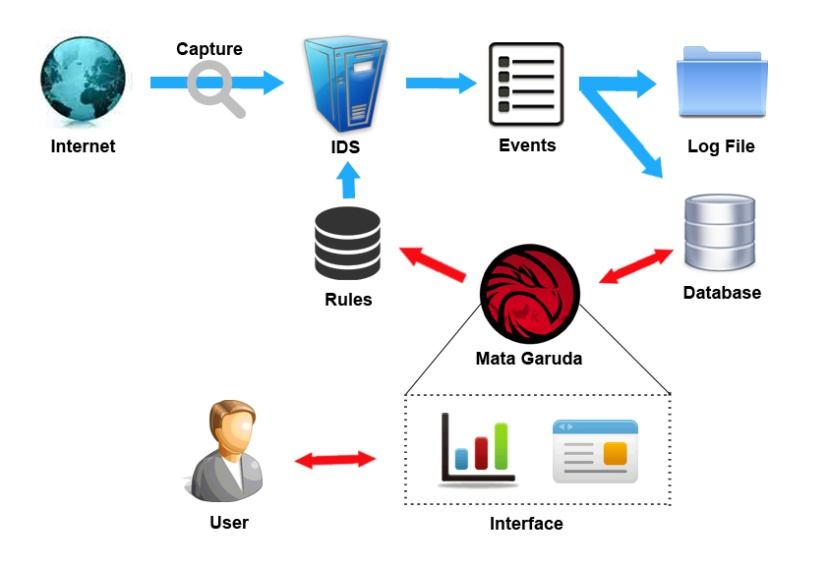
## **Deskripsi Umum Sistem**

Pada proyek akhir ini penulis melakukan perancangan dan pembangunan Aplikasi Mata Garuda. Aplikasi ini merupakan suatu aplikasi antar muka berbasis *web application* yang dapat menyajikan data yang dihasilkan oleh sistem *Intrution Detection System* (IDS) *open source* ke dalam bentukaplikasi web yang lebih informatif. Aplikasi ini dirancang dan dibangun untuk memudahkan administrator jaringan dalam memantau dan melakukan analisa terhadap *event-event* atau bentuk aktifitas yang terjadi pada jaringan yang di-*monitoring* baik yang sifatnya berguna ataupun berdampak buruk bagi suatu jaringan komputer beserta komponennya*.*

Data *events* yang dihasilkan oleh IDS berbasis *open source* berisi berbagai informasi yang terdapat pada *IP Header* sebuah paket yang melalui jaringan komputer,yaitu: *Source IP Address, Destination IP Address, Source Port, Destination Port,* jenis *events,* waktu, *payload,* dan sebagainya. Data-data tersebut terlebih dahulu disimpan oleh IDS berbasis *open source* ke dalam *database* yang nantinya akan diintegrasikan dengan aplikasi Mata Garuda.

Aplikasi Mata Garuda akan menyajikan data-data hasil *monitoring* IDS yang tersimpan pada *database* ke bentuk yang lebih variatif berupa statistik *events,* chart, informasi letak *sensor* IDS, hingga *payload* data yang diterejemahkan kedalam format ASCII dikarenakan data yang dihasilkan oleh IDS masih dalam bentuk *hexadecimal.*

Gambaran umum dari aplikasi proyek akhir ini dapat dilihat pada gambar berikut



Pada gambar terdapat dua alur berbeda (alur yang ditandai dengan panah biru dan alur yang ditandai dengan panah merah) yang menjelaskan proses yang berbeda pula. Alur warna biru merupakan proses sistem IDS memantau jaringan secara keseluruhan dan alur warna merah merupakan proses aplikasi mengambil dan menyajikan data.

Berikut penjelasan untuk setiap alur yang ada pada gambaran umum sistem:

1. Alur sistem IDS (Alur Biru)

Alur data dimulai dengan sistem IDS (yang telah dikonfigurasi sebelumnya) yang melakukan *monitoring* jaringan internet dan melakukan *capture packet* yang melintasi jaringan yang di-*monitoring. Packet* yang berhasil di-*capture* tersebut akan disesuaikan terlebih dahulu dengan *rules* yang terdapat pada *repositoty* IDS, jika terdapat *matching pattern* antara *packet* dan *rules* maka *packet* tersebut dikategorikan sebagai sebuah *events* yang memiliki tingkat bahaya tertentu. Tugas IDS yang menentukan apakah prioritas atau tingkatan bahaya *events* tersebut tinggi atau tidak berdasarkan *rules* yang ada. Setelah mengkategorikan *events,* selanjutnya *events* tersebut akan disimpan ke dalam *log file* IDS (konfigurasi standar) dan juga *database* IDS (konfigurasi sesuai kebutuhan).

1. Alur Aplikasi (Alur Merah)

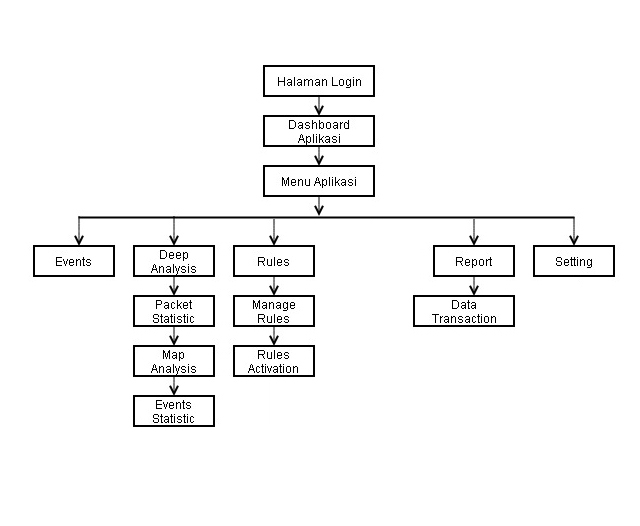
Pada Alur ini aplikasi Mata Garuda akan mengambil semua data yang terdapat pada *database* IDS untuk kemudian mengolahnya sehingga dapat disajikan kedalam bentuk yang lebih informatif dan variatif. Data yang diambil hanya data yang terdapat pada *database* sementara data yang terdapat pada *log file* tidak dibahas pada Proyek Akhir ini. Pada gambar terlihat bahwa Aplikasi Mata Garuda berinteraksi dengan *database* secara *vice versa,* dalam artian bahwa *database* memberikan semua data yang berasal dari sistem IDS ke aplikasi namun dari sisi aplikasi yang dioperasikan oleh administrator juga dapat melakukan pengubahan dan penambahan data pada *database.* Data yang diubah atau ditambahkan pada *databae* hanya sebatas data yang berhubungan dengan konfigurasi aplikasi dan bukan data hasil *capture* IDS. Aplikasi Mata Garuda juga berinteraksi dengan *rules repository* IDS, dimana administrator diberi kemudahan untukdapat mengubah dan menambahkan *file rules* pada IDS. Alur terakhir yaitu hubungan *vice versa* antara pengguna dan *interface* aplikasi dimana pengguna dapat melihat antara lain grafik dari data yang didapat pada *database* dan tampilan menu antar muka lainnya.

## **Perancangan Sistem**

Pada bagian ini akan dijelaskan lebih detail mengenai rancangan dari aplikasi yang akan dibangun, dan menjadi landasan bagi tahap selanjutnya yaitu Pengembangan Aplikasi.

### **Rancangan Aplikasi**

Pada aplikasi Mata Garuda ini terdapat beberapa *modul* dan fungsi yang saling terkait satu dengan yang lainnya namun tetap dapat dikembangkan secara mandiri dan terpisah, sesuai dengan konsep *modularity* yang sangat ditekankan oleh penulis. Proyek Akhir ini hanya mencakup *core sytem* dari aplikasi Mata Garuda yang hanya menampilkan data pada *database* ke antar muka aplikasi, untuk modul dan fungsi lainnya (*Deep Analysis: Packet Statistic, Map Analysis; Rules: Manage Rules, Rules Activation*) dibangun dan dikembangkan secara terpisah dan mandiri. Rancangan tersebut dapat dijelaskan melalui diagram blok seperti gambar berikut:



Berikut ini penjelasan untuk blok diagram diatas:

1. Halam *Login*

Suatu aplikasi yang baik terutama aplikasi berbasis *web application* membutuhkan sebuah autentikasi pengguna untuk dapat mengakses aplikasi dan menggunakannya. Halaman *login* ini dibuat dengan tujuan agar orang-orang yang diberi tanggung jawab sebagai seroang administrator jaringan yang dapat mengakses aplikasi.

1. *Dashboard* Aplikasi

Merupakan tampilan awal aplikasi setelah seorang administrator berhasil masuk ke dalam sistem. *Dashboard* hanya menampilkan data kesimpulan dari sistem dan bukan detail data, data yang ditampilkan berupa *real time* data, top *events,* sensor, dan juga jumlah dan tingkatan bahaya dari *events* yang tersimpan.

1. Menu Aplikasi

Terdiri dari beberapa menu untuk menuju ke beberapa fungsi yang terdapat pada aplikasi Mata Garuda yaitu: *Events*, *Deep* *Analysis*, *Rules*, *Report*, *Setting*.

* 1. Menu *Events*

Pada menu *events,* menampilkan seluruh data event yang tersimpan pada *database* setelah dilakukan relasi antar tabel untuk menyajikan data yang dibutuhkan oleh administrator. Kategori dari setiap *events* yang ditampilkan cukup banyak sehingga diperlukan sebuah fungsi atau *action* tersendiri yang dapat menyajikan data detail dari masing-masing *events.*

* 1. Menu *Deep Analysis*

Merupakan *modul* atau fungsi pada aplikasi Mata Garuda untuk melakukan analisa terhadap paket yang berhasil dipantau. Menu *Deep Analysis* terdiri dari 3 submenu yaitu:

1. *Packet Statistic*: Menampilkan statistik dari paket yang melintas di dalam jaringan dalam bentuk *pie chart* yang menjelaskan *source/destination IP address, source/destination Port,* beserta jumlah paket data yang berhasil dipantau.
2. *Maps Analysis*: Menampilkan peta dunia dimana pada peta tersebut terdapat beberapa titik lokasi yang menandakan *source/destination IP Address* sesuai dengan kriteria yang dimasukkan oleh administrator.
3. *Events Statistic*: Menampilkan grafik dan *chart* dari banyaknya *events* yang tersimpan pada *database*. Grafik untuk jumlah *events* keseluruhan dan *pie chart* untuk menunjukan banyaknya *protocol* yang berhasil dipantau. Grafik dan *chart* yang disajikan sesuai dengan kriteria dan masukkan dari administrator, baik berupa waktu kejadian *events* ataupun bisa melalui banyakknya *sensor*.
   1. Menu *Rules*

Salah satu keunggulan dari Mata Garuda daripada aplikasi antar muka IDS yang lainnya adalah mampu menyediakan antar muka untuk melakukan perubahan *rules* dari IDS yang digunakan. *Rules* biasanya secara manual diubah oleh administrator langsung ke dalam sistem IDS dengan melakukan beberapa konfigurasi. Menu *Rules* terdiri dari 2 submenu yaitu:

1. *Manage Rules*: Memberikan kemudahan bagi administrator untuk membuat dan melakukan perubahan *rules* tanpa harus berhubungan dengan sistem atau mengubah konfigurasi sistem secara langsung.
2. *Activation Rule*: Fungsi ini berguna untuk mengaktifkan atau menonaktifkan *rules* yang ada pada sebuah sensor.
   1. Menu *Report*

Menampilkan data transaksi dari *events* yang ada. Transaksi dapat ditampilkan berdasarkan hari, minggu, bulan, ataupun tahun.

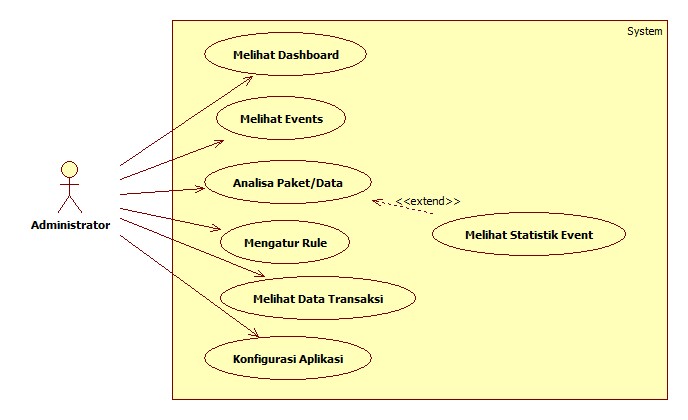
* 1. Menu *Setting*

Pada menu ini terdapat beberapa fungsi untuk melakukan perubahan *IP Address sensor* yang dipantau dan juga menentukan *threshold* untuk *events.*

### **Use Case Diagram**

*Use Case Diagram* adalah sebuah diagram yang merepresentasikan interaksi antara pengguna dan sistem dimana pada penggambarannya menerapkan langkah-langkah yang dapat dilakukan oleh pengguna terhadap suatu sistem. *Use Case* berguna untuk memberi penjelasan mengenai fungsi-fungsi yang ada pada sistem dan langsung berinteraksi dengan manusia.

Berikut ini merupakan *Use Case Diagram* dari aplikasi Mata Garuda yang akan dibangun:



*Use Case diagram* pada gambar menjelaskan interaksi yang dapat dilakukan pengguna dengan sistem dalam hal ini aplikasi Mata Garuda.

1. **Melihat *Dashboard***

Administrator berhasil masuk ke sistem, halaman pertama yang diberikan oleh aplikasi Mata Garuda yaitu tampilan *dashboard* aplikasi. Administrator dapat melihat data *real time* dari serangan yang terjadi per satuan waktu, jumlah sensor yang dimiliki IDS beserta status konektivitasnya, *top* 10 kategori *event* (berdasarkan *source/destination IP address* dan juga *protocol/port* yang digunakan), hingga jumlah *event* yang terjadi untuk tiap-tiap tingkatan bahaya (*severity: high, medium, low*).

1. **Melihat *Events***

Aplikasi menyediakan menu atau fungsi yang dapat menampilkan keseluruhan *events* yang telah tersimpan ke dalam *database.* Administrator dapat melihat *events* dengan cara memilih menu *event* dan masuk ke halaman *events.* Administrator kemudian melihat informasi dari masing-masing *event* yang ada, mulai dari *severity event, source IP address, destination IP address, event signature, timestamp,* hingga *payload* yang dihasilkan oleh setiap *event.* Data ditampilkan dalam bentuk tabel dan terdapat menu detail untuk setiap *event* untuk melihat informasi lebih lengkap lagi dari *event* tersebut. Pada halaman *event* administrator juga dapat memasukkan kriteria pencarian berdasarkan waktu, *IP address, port,* dan sebagainya sehingga data yang ditampilkan atau disajikan hanya yang mengandung kriteria yang dimasukkan oleh administrator.

1. **Analisa Paket/Data**

Bagian yang diunggulkan pada aplikasi Mata Garuda ini yaitu memiliki menu analisa paket/data, dimana paket/data yang dimaksud adalah yang berhasil ditangkap oleh *sensor* IDS karena tidak sesuai dengan *rules* yang telah didefenisikan sebelumnya. Memiliki 3 submenu yaitu: Packet Statistic, Maps Analysis, dan Events Statistic. Namun yang dikembangkan oleh penulis sesuai dengan studi kasus Proyek Akhir (*core system*) hanya pada Events Statistic. Untuk Packet Statistic dan Maps Analysis dikembangkan oleh tim lainnya.

Pada menu analisa paket/data, nantinya administrator dapat melakukan analisa terhadap suatu *trend events* sebagai bagian dari proses *forensic network,* data diberikan cukup lengkap sehingga semakin memudahkan administrator untuk mengambil keputusan. Pada Proyek Akhir ini (pengembangan *core system* Mata Garuda) penulis akan menampilkan Events Statistic pada submenu Analisa Paket, dimana data yang ditampilkan berupa grafik garis yang menunjukan jumlah *events* yang berhasil didapat perjam/hari. Dari grafik yang dimunculkan, administrator akan mengambil kesimpulan pada jam berapa serangan ke jaringan internet paling banyak dilakukan setiap harinya. Administrator juga dapat memasukkan kriteria pemilihan hari, untuk membandingkan data pada hari yang berbeda.

Selain grafik garis, penulis juga menampilkan grafik lain berbentuk *pie chart* dimana pada grafik ini data yang ditampilkan yaitu jumlah dan presentase dari masing-masing *protocol* yang ada pada jaringan (*protocol* ICMP, TCP, dan UDP).

1. **Mengatur *Rule***

Rule merupakan salah satu bagian penting pada IDS berbasis *open source, rule* yang terdapat pada IDS yang menentukan apakah sebuah paket data yang melintas di dalam jaringan masuk ke dalam kategori paket berbaha atau tidak. Pada aplikasi Mata Garuda, administrator diberi kemudahan untuk dapat melakukan pengaturan terhadap *rule-rule* yang ada pada masing-masing *sensor* IDS tanpa harus berhubungan langsung dengan sistem IDS. Hal ini juga akan memberikan kemudahan jika *sensor* IDS yang diatur oleh administrator lebih dari satu *sensor,* dalam artian semua konfigurasi *rule* untuk tiap-tiap *sensor* IDS terpusat pada aplikasi Mata Garuda.

1. **Melihat Data Transaksi / *Report***

Administrator dapat melihat data transaksi dari keseluruhan *event* yang telah berhasil tersimpan ke dalam *database.* Data transaksi ini berguna sebagai sebuah *report* atau laporan.

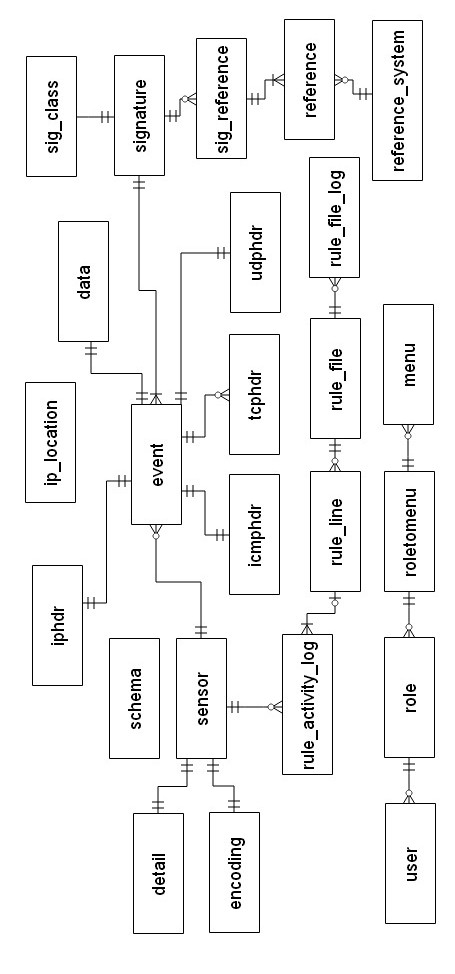
1. **Konfigurasi Aplikasi**

Aplikasi Mata Garuda menyediakan beberapa konfigurasi terhadap IDS, diantaranya ialah memperbolehkan administrator untuk melakukan pengubahan *IP address sensor* IDS.

### ***Entity Relationship Diagram***

*Entitiy Relationship Diagram* (ERD) adalah suatu representasi grafik dari beberapa entitas dan hubungan tiap-tiap entitas yang ada dalam bentuk relasi. Biasanya digunakan dalam dunia IT untuk melakukan dan mengorganisasi data di dalam sebuah *database* yang terdapat pada suatu aplikasi atau sistem informasi, dimana entitas merupakan bentuk nyata (biasa disebut dengan tabel) sebagai tempat dimana data disimpan.

Pada proyek akhir ini penulis merancang ERD yang ditujukan untuk pembangunan dan pengembangan aplikasi antar muka IDS Mata Garuda, seperti yang dijelaskan pada gambar.



Gambar diatas menunjukan rancangan *database* aplikasi Mata Garuda bersama dengan tabel-tabel relasi yang ada.

Berikut ini detail untuk masing-masing tabelyang ada:

1. **Deskripsi Tabel event**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* | *Pri* | *Sensor ID* |
| *cid* | *bigint* | *Pri* | *Event ID* |
| *signature* | *int* | *Mul* | *Signature ID* |
| *timestamp* | *datetime* | *Mul* | *Time of events logged* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *sensor (sid)*  *iphdr (sid,cid)*  *icmphdr (sid,cid)*  *tcphdr (sid,cid)*  *udphdr (sid,cid)*  *data (sid,cid)*  *signature (signature)* |
| Deskripsi | : | Tabel *event* berisi semua *event* yang berhasil disimpan oleh IDS Snort dan memiliki waktu terjadinya *event* |

1. **Deskripsi Tabel sensor**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* | *Pri* | *Sensor ID* |
| *hostname* | *text* |  | *Hostname Sensor* |
| *interface* | *text* |  | *Network interface* |
| *filter* | *text* |  | *BPF filter* |
| *detail* | *tinyint* |  | *Detail level logging* |
| *encoding* | *tinyint* |  | *Encoding format of the payload* |
| *last\_cid* | *bigint* |  | *Last event ID* |
| *ip\_address* | *inet* |  | *IP address sensor* |
| *location* | *varchar* |  | *Sensor location* |
| *company* | *varchar* |  | *Sensor owner* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid)*  *detail (detail)*  *encoding (encoding)* |
| Deskripsi | : | Tabel *sensor* berisi informasi mengenai *sensor* yang terhubung dengan *database* |

1. **Deskripsi Tabel detail**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *detail\_type* | *smallint* | *Pri* | *Detail type* |
| *detail\_text* | *text* |  | *Detail info* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *sensor (detail\_type)* |
| Deskripsi | : | Detail dari tabel *sensor* |

1. **Deskripsi Tabel encoding**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *encoding\_type* | *smallint* | *Pri* | *Encoding type* |
| *encoding\_text* | *text* |  | *Encoding info* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *sensor (encoding\_type)* |
| Deskripsi | : | *Encoding* dari tabel *sensor* |

1. **Deskripsi Tabel iphdr**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* | *Pri* | *Sensor ID* |
| *cid* | *bigint* | *Pri* | *Event ID* |
| *ip\_src* | *int* | *Mul* | *Source IP Address* |
| *ip\_dst* | *int* | *Mul* | *Destination IP Address* |
| *ip\_ver* | *tinyint* |  | *Ip version* |
| *ip\_hlen* | *tinyint* |  | *Ip header length* |
| *ip\_tos* | *tinyint* |  | *Ip type of service* |
| *ip\_len* | *smallint* |  | *IP datagram length* |
| *ip\_id* | *smallint* |  | *IP ID* |
| *ip\_flags* | *tinyint* |  | *IP flags* |
| *ip\_off* | *smallint* |  | *IP fragment offset* |
| *ip\_ttl* | *Tinyint* |  | *IP time to live* |
| *ip\_proto* | *tinyint* |  | *IP protocol* |
| *ip\_csum* | *smallint* |  | *IP checksum* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *iphdr* berisi semua informasi dari *iphdr* sebuah *event* |

1. **Deskripsi Tabel icmphdr**

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *icmphdr* memuat informasi dari *protocol* ICMP *event* |

1. **Deskripsi Tabel tcphdr**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* | *Pri* | *Sensor ID* |
| *cid* | *bigint* | *Pri* | *Event ID* |
| *tcp\_sport* | *smallint* | *Mul* | *TCP source port* |
| *tcp\_dport* | *smallint* | *Mul* | *TCP destination port* |
| *tcp\_seq* | *int* |  | *TCP sequence number* |
| *tcp\_ack* | *int* |  | *TCP ACK number* |
| *tcp\_off* | *tinyint* |  | *TCP offset* |
| *tcp\_res* | *tinyint* |  | *TCP reserved* |
| *tcp\_flags* | *tinyint* |  | *TCP flags* |
| *tcp\_win* | *smallint* |  | *TCP window* |
| *tcp\_csum* | *smallint* |  | *TCP checksum* |
| *tcp\_urp* | *smallint* |  | *TCP urgent pointer* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *tcphdr* memuat informasi dari *protocol* TCP *event* |

1. **Deskripsi Tabel udphdr**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* | *Pri* | *Sensor ID* |
| *cid* | *bigint* | *Pri* | *Event ID* |
| *udp\_sport* | *smallint* | *Mul* | *UDP source port* |
| *udp\_dport* | *smallint* | *Mul* | *UDP destination port* |
| *udp\_len* | *smallint* |  | *UDP length* |
| *udp\_csum* | *smallint* |  | *UDP checksum* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *udphdr* memuat informasi dari *protocol* UDP *event* |

1. **Deskripsi Tabel data**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* | *Pri* | *Sensor ID* |
| *cid* | *bigint* | *Pri* | *Event ID* |
| *data\_payload* | *text* |  | *Packet payload* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Berisi tentang informasi data *payload* untuk setiap *event* |

1. **Deskripsi Tabel signature**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sig\_id* | *int* | *Pri* | *Signature ID* |
| *sig\_name* | *varchar* | *Mul* | *Signature name* |
| *sig\_class\_id* | *int* |  | *Classification ID* |
| *sig\_priority* | *int* |  | *Priority* |
| *sig\_rev* | *int* |  | *Revision number* |
| *sig\_sid* | *int* |  | *Internal signature id* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sig\_id)*  *sig\_class (sig\_class\_id)*  *sig\_reference (sig\_id)* |
| Deskripsi | : | Tabel *signature* merupakan salah satu tabel utama pada *database* IDS Snort, yang berisi semua *signature known threats* |

1. **Deskripsi Tabel sig\_class**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sig\_class\_id* | *int* | *Pri* | *Signature classification ID* |
| *sig\_class\_name* | *varchar* | *Mul* | *Classification name* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *signature (sig\_class\_id)* |
| Deskripsi | : | Berisi klasifikasi dari *signature* yang ada |

1. **Deskripsi Tabel sig\_reference**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sig\_id* | *int* | *Pri* | *Signature ID* |
| *ref\_seq* | *int* | *Pri* | *Reference sequence number* |
| *ref\_id* | *int* |  | *Reference ID* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *signature (sig\_id)*  *reference (ref\_id)* |
| Deskripsi | : | Tabel informasi *signature reference* |

1. **Deskripsi Tabel references**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *ref\_id* | *int* | *Pri* | *Reference ID* |
| *ref\_system\_id* | *int* |  | *Referemce system ID* |
| *ref\_tag* | *varchar* |  | *Reference tag* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *sig\_reference (ref\_id)*  *reference\_system (ref\_system\_id)* |
| Deskripsi | : | Tabel *reference* dari *vendor* |

1. **Deskripsi Tabel reference\_system**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *ref\_system\_id* | *int* | *Pri* | *Reference system ID* |
| *ref\_system\_name* | *varchar* |  | *Reference system name* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *reference(ref\_system\_id)* |
| Deskripsi | : | Tabel sistem *reference* |

1. **Deskripsi Tabel ip\_location**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *ip\_start* | *inet* | *Pri* | *Start IP address* |
| *ip\_end* | *inet* | *Pri* | *End IP address* |
| *country* | *text* |  | *Country of IP address* |
| *stateprov* | *text* |  | *State of IP address* |
| *city* | *text* |  | *City of IP address* |
| *latitude* | *double* |  | *Latitude* |
| *longitude* | *double* |  | *Longitude* |
| *timezone\_offset* | *varchar* |  | *Timezone offset* |
| *timezone\_name* | *text* |  | *Timezone name* |
| *isp\_name* | *text* |  | *Internet Service Provider name* |
| *connection\_type* | *text* |  | *Connection type* |
| *organization\_name* | *text* |  | *Organization owner* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | - |
| Deskripsi | : | Tabel *ip\_location* memuat semua daftar *ip address* yang ada di dunia beserta informasinya. Tabel ini tidak berelasi dengan tabel manapun, namun penggunaanya akan dibutuhkan pada fungsi aplikasi Mata Garuda. |

1. **Deskripsi Tabel rule\_activity\_log**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *id\_activity\_log* | *varchar* | *Pri* | *ID log activity* |
| *id\_line* | *varchar* |  | *Line ID* |
| *sensor* | *varchar* |  | *Sensor name* |
| *actived* | *boolean* |  | *Rule state* |
| *modify* | *timestamp* |  | *Time of modify* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *rule\_line (id\_line)* |
| Deskripsi | : | Tabel yang memuat semua aktifitas *rule* yang dibuat oleh administrator |

1. **Deskripsi Tabel rule\_line**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *id\_line* | *varchar* | *Pri* | *Line ID* |
| *id\_file* | *varchar* |  | *File ID* |
| *action* | *char* |  | *Action rule* |
| *msg* | *text* |  | *Message rule* |
| *line* | *text* |  | *Line rule* |
| *sid* | *varchar* |  |  |
| *gid* | *varchar* |  |  |
| *rev* | *varchar* |  | *Revision* |
| *created* | *timestamp* |  | *Date of created* |
| *author* | *varchar* |  | *Rule author* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *rule\_activity\_log (id\_line)*  *rule\_file (id\_file)* |
| Deskripsi | : |  |

1. **Deskripsi Tabel rule\_file**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *file\_name* | *varchar* | *Pri* | *Rule file name* |
| *full\_path* | *text* |  | *Rule full path* |
| *info* | *text* |  | *Rule info* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *rule\_line (file\_name)*  *rule\_file\_log (id\_file)* |
| Deskripsi | : | Tabel *rule\_file* memuat informasi nama file dan *path* file pada server |

1. **Deskripsi Tabel rule\_file\_log**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *id\_file\_log* | *varchar* | *Pri* | *File ID log* |
| *id\_file* | *varchar* |  | *File ID* |
| *size* | *integer* |  | *Rule log size* |
| *modify* | *timestamp* |  | *Date of modify* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *rule\_file (file\_name)* |
| Deskripsi | : |  |

1. **Deskripsi Tabel user**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *userid* | *integer* | *Pri* | *ID user* |
| *roleid* | *integer* |  | *ID role* |
| *username* | *varchar* |  | *Username* |
| *userpassword* | *varchar* |  | *Password user* |
| *userfullname* | *varchar* |  | *Full name* |
| *useraddress* | *text* |  | *Address user* |
| *userphone* | *varchar* |  | *Phone user* |
| *useractive* | *boolean* |  | *User status* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *role (roleid)* |
| Deskripsi | : | Tabel *user* berisi semua informasi bagi administrator yang berhak mengakses aplikasi |

1. **Deskripsi Tabel role**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *roleid* | *integer* | *Pri* | *Role ID* |
| *rolename* | *varchar* |  | *Role name* |
| *roledescription* | *varchar* |  | *Role description* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *user (role\_id)*  *roletomenu (role\_id)* |
| Deskripsi | : | Tabel yang memuat informasi *role* dari *user* yang ada pada *database* |

1. **Deskripsi Tabel roletomenu**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *menuid* | *integer* | *Pri* | *Menu ID* |
| *roleid* | *integer* | *Pri* | *Role ID* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *role (role\_id)*  *menu (menu\_id)* |
| Deskripsi | : | Menentukan *role* untuk berhak mengakses *menu* |

1. **Deskripsi Tabel menu**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *menuid* | *integer* | *Pri* | *Menu ID* |
| *menuname* | *varchar* |  | *Menu name* |
| *menudescription* | *text* |  | *Menu description* |
| *menualias* | *varchar* |  | *Menu alias* |
| *menuicon* | *varchar* |  | *Menu icon* |
| *menucontroller* | *varchar* |  | *Menu controller* |
| *menulink* | *varchar* |  | *Menu link* |
| *menuorder* | *integer* |  | *Menu order* |
| *menulevel* | *integer* |  | *Menu level* |
| *menuparent* | *integer* |  | *Menu parent* |
| *menutype* | *integer* |  | *Menu type* |
| *menuactive* | *boolean* |  | *Menu status* |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *roletomenu (menuid)* |
| Deskripsi | : | Tabel *menu* memuat semua informasi *menu* yang akan ditampilkan pada aplikasi Mata Garuda |

1. **Deskripsi Tabel schema**

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *vseq* | *integer* | *Pri* | *Version sequence* |
| *ctime* | *timestamp* |  | *Create time* |

## **Kebutuhan Pendukung Pengembangan**

### **Kebutuhan Perangkat Keras**

Perangkat keras yang digunakan pada pengembangan aplikasi antar muka IDS Snort Mata Garuda ditujukan untuk *server* dan juga *client* yang bertindak sebagai *sensor,* sehingga spesifikasi untuk *server* dan *client* adalah sama. Berikut ini merupakan tabel spesifikasi dari perangkat keras yang dibutuhkan:

|  |  |
| --- | --- |
| Processor | Inter Core i7 |
| Memory | 32 GB DDR3 2800MHz performance memory |
| Hard Drives | SSD (1TB) and HDD (1TB) |
| Integrated PCI adapted slots | Minimum 6 PCIs slots |

### **Kebutuhan Perangkat Lunak**

Perangkat lunak yang digunakan mencakup sistem operasi, hingga aplikasi-aplikasi atau *service* yang dibutuhkan untuk pengembangan. Berikut ini merupakan tabel perangkat lunak yang dibutuhkan:

|  |  |
| --- | --- |
| Oparating System | Debian 7 |
| Sensor IDS | Snort |
| DBMS | PostgreSQL |
| Web Service | Apache, PHP, JavaScript |
| Frameworks | CodeIgniter |
| Editor | Sublime, Edit Plus 3 |
| Virtualization | VMWare, Oracle VirtualBox |

1. **IDS Snort**

IDS Snort seperti yang telah dijelaskan pada subbab 2.5 bertindak sebagai sistem *monitoring* yang menangkap *packet* pada jaringan dan menyesuaikan *packet* tersebut dengan *rules* yang telah didefinisikan kemudian menyimpannya ke dalam database. IDS Snort akan dikonfigurasi sesuai dengan kebutuhan administrasi jaringan.

1. **PostgreSQL**

PostgreSQL seperti yang telah dijelaskan pada subbab 2.6 dibutuhkan sebagai media atau sistem penyimpanan dari setiap *sensor* untuk menyimpan hasil *monitoring-*nya. Pada aplikasi Mata Garuda nantinya, desain *database* yang telah dirancang akan dibangun menggunakan DBMS PostgreSQL ini.

1. **PHP**

Merupakan salah satu bahasa pemrograman yang paling banyak digunakan saat ini terutama untuk aplikasi-aplikasi yang berbasis *web.* PHP merupakan bahasa pemrograman *server-side scripting* yang didesain bagi kepentingan pengembangan aplikasi berbasis *web* namun tidak tertutup kemungkinan untuk pengembangan pada bidang umum lainnya. Code PHP juga dapat digabungkan dengan bahasa pemrograman lainnya seperti HTML dan juga JavaScript.

Pada pengembangan aplikasi Mata Garuda ini, PHP merupakan bahasa pemrograman utama dan menjadi salah satu keunikan dari Mata Garuda.

1. **JavaScript**

Merupakan salah satu bahasa pemrograman komputer yang bersifat dinamis. Banyak digunakan sebagai bagian dari *web-*browseryang dalam penerapannya memungkinkan *client-side script,* interaksi dengan pengguna, mengatur *browser,* dan mengubah isi konten menjadi tampilan yang lebih variatif. Fungsi-fungsi yang terdapat ada JavaScript tersebut berguna untuk aplikasi Mata Garuda dalam hal tampilan dan *layout* penyajian data/informasi kepada pengguna melalui *web-browser.*

1. **CodeIgniter**

*Framework* PHP yang digunakan penulis untuk mengembangkan aplikasi Mata Garuda ialah CodeIgniter, satu dari banyak *framework* yang beredar saat ini. Pengembangan aplikasi Mata Garuda akan menyesuaikan dengan direktori *framework* CodeIgniter namun juga dilakukan beberapa penambahan pada beberapa sektor seperti *core controller* dan juga *library* CodeIgniter tersebut.

1. **PHP Editor**

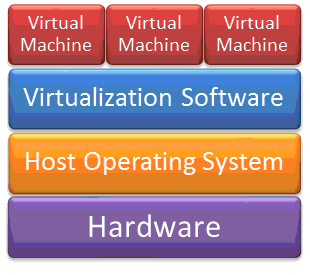
Banyak sekali PHP *editor* yang dapat digunakan dewasa ini, para pengembangan aplikasi *web* pun mempunyai alasan masing-masing dalam memilih dan menggunakan PHP *editor* tersebut. Pada proyek akhir ini penulis menggunakan dua PHP *editor* yaitu Sublime dan Edit Plus 3. Alasan utama ialah kemudahan fungsi yang diberikan dan juga lebih familiar terhadap dua *editor* tersebut, pada proyek akhir ini penulis tidak membahas lebih dalam alasan pemilihan kedua editor tersebut.

1. **Virtual Machine**

Virtual Machine merupakan salah satu sistem yang banyak dipakai dalam dunia komputer untuk membagi-bagi sistem operasi yang dapat berjalan sendiri-sendiri namun secara fisik masih berada pada perangkat yang sama, dengan artian secara logik suatu komputer bisa memiliki banyak sistem operasi yang bisa dijalankan secara bersama-sama ataupun tidak. Virtual Machine beroperasi berdasarkan arsitektur komputer beserta fungsi-fungsi yang dimiliki dan pada penerapannya sering kali juga membutuhkan pemahaman mengenai perangkat keras dan perangkat lunak secara *virtual.*

Pada proyek akhir IDS *open source* Snort diletakkan pada mesin yang berbeda dengan database dan aplikasi Mata Garuda, hal ini dikarenakan IDS Snort akan memantau banyak jaringan dan tentunya membutuhkan banyak komputer dimana komputer-komputer IDS tersebut disebut sebagai komputer *sensor* dan komputer yang memuat *database* dan aplikasi Mata Garuda disebut komputer *defense centre.* Dari permasalahan tersebut, penulis menerapkan Virtual Machine dalam pengembangan aplikasi Mata Garuda sebagai bagian dari efesiensi biaya dan waktu sebelum pada akhirnya akan diterapkan pada topologi yang nyata dan bukan virtual. Virtual Machine yang digunakan yaitu VM Ware dan juga Oracle VirtualBox (khusus untuk personal computer penulis).

Virtual Machine sama seperti *software* lainnya yaitu tetap berjalan di dalam sebuah sistem operasi seperti yang terdapat pada gambar

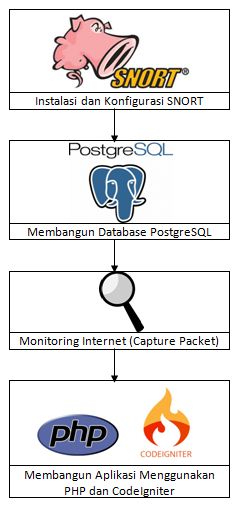


Dari gambar dapat dijelaskan bahwa sebuah *virtualization software* berjalan di atas (di dalam) sebuah sistem operasi, baik Windows, Linux, ataupun sistem operasi lainnya. *Virtualization Software* tersebut dapat memuai satu atau lebih *virtual machine* yang dapat berjalan sama baiknya.

## **Pengembangan Aplikasi**

Pada bagian ini menjelaskan mengenai pembuatan dan pengembangan sistem melalui rancangan yang telah dibuat sebelumnya, serta implementasi yang dilakukan terhadap sistem.

Aplikasi Mata Garuda ini dibuat melalui beberapa proses, mulai dari instalasi dan konfigurasi IDS Snort untuk menghasilkan data *monitoring* hingga pengembangan menggunakan *framework* PHP Codeigniter, seperti yang dijelaskan seperti gambar:



Gambar diatas merupakan gambar desain implementasi dari aplikasi antar muka IDS Snort, Mata Garuda. Penjelasan untuk tiap tapahan pengembangan diatas berada pada subbab selanjutnya.

### **Instalasi dan Konfigurasi Snort**

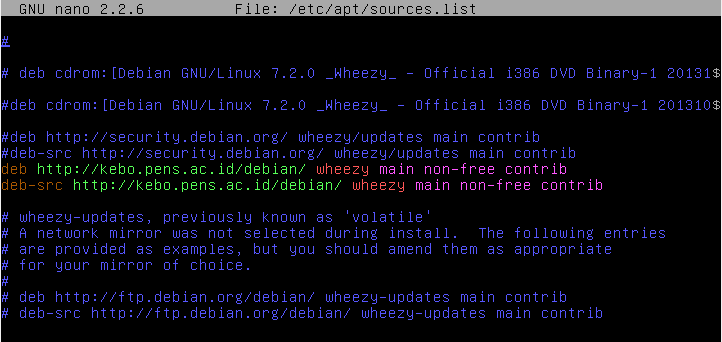
Pada Proyek Akhir pembangunan aplikasi antar muka IDS Snort Mata Garuda ini penulis bekerja pada *environment* Linux, sehingga proses instalasi dan konfigurasi *service-service* yang dibutuhkan merupakan proses yang hanya terdapat pada *environment* Linux. Berikut ini proses instalasi dan konfigurasi Snort pada sistem operasi Linux distribusi Debian 7.

* 1. Intalasi Snort

Untuk dapat melakukan instalasi *package* atau *service* pada Linux, terlebih dahulu harus masuk sebagai *root privileges.* Langkah-langkah dibawah ini merupakan cara untuk instalasi *package* Snort pada sistem operasi Debian 7.

* 1. Cek daftar repositori yang terdapat pada konfigurasi Debian:

|  |
| --- |
| root#vi /etc/apt/source.list |



Pada gambar diatas, penulis menggunakan repositori PENS (kebo.pens.ac.id). Masih banyak daftar repositori yang terdapat pada regional Indonesia contohnya UI (kambing.ui.ac.id).

* 1. Setelah repositori diatur, lakukan pengecekan apakah repositori tersebut tersedia atau tidak dengan cara:

|  |
| --- |
| root#apt-get update |

* 1. Instal *package* yang diperlukan, dalam hal ini yaitu Snort dan juga PostgreSQL (PostgreSQL hanya sebatas instalasi dan konfigurasi, bukan desain *database*)

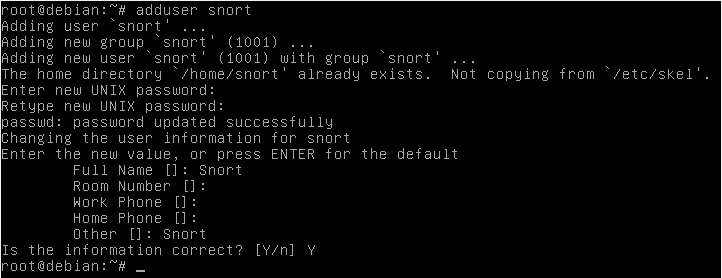
|  |
| --- |
| root#apt-get install snort postgresql snort-pgsql |

* 1. Konfigurasi Snort

Jika *package-package* yang dibutuhkan telah terinstal dengan baik, saatnya untuk melakukan konfigurasi Snort. Snort perlu dikonfigurasi agar sesuai dengan kebutuhan administrator jaringan.

1. Buat user Snort pada *lingkungan* *root*:

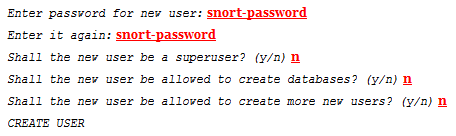
|  |
| --- |
| root#deluser snort  root#adduser snort |



Isikan masukkan yang sesuai pada saat pembuatan user snort yang baru.

1. Masuk ke lingkungan *user* postgres (*user* postgres otomatis tercipta ketika *package* postgresql diinstal ke sistem)

|  |
| --- |
| root#su postgres  postgres$createdb snort  postgres$zcat /usr/share/doc/snort-pgsql/create\_postgresql.gz | psql snort  postgres$createuser -P snort |



*User* snort diatas berbeda dengan *user* snort yang telah dibuat sebelumnya pada *environment root, user* snort diatas bertindak sebagai *schema* untuk *database* snort. Perintah “zcat” ialah sebagai instalasi *database* awal dari IDS Snort. *Database* tersebut nantinya akan dimodifikasi sesuai dengan kebutuhan pengembangan aplikasi dan sesuai rancangan ERD.

1. Masuk ke *database* snort untuk memberikan semua *privileges* pada *database* ke *schema* snort yang telah dibuat sebelumnya

|  |
| --- |
| postgres$psql snort  psql>grant all privileges on database snort to snort; |

1. Lakukan konfigurasi pada file snort.conf, untuk menghubungkan IDS dengan *database* yang diinginkan

|  |
| --- |
| root#vi /etc/snort/snort.conf |

Ubah beberapa masukkan pada baris berikut :

|  |
| --- |
| output database: alert, postgresql, user=snort password=snort-password dbname=snort host=postgresql-host-ip sensor\_name=sensor01 |

*user* : snort (*schema* yang telah dibuat sebelumnya)

*password* : *password* dari *database* snort

*dbname* : snort (*database* yang telah dibuat sebelumnya)

*host* : IP *address* atau *hostname* dari *database* tujuan

*sensor\_name* : sensor01 (nama sensor sesuai kebutuhan)

Khusus untuk sistem operasi Debian, konfigurasi *database* terdapat pada database.conf

|  |
| --- |
| root#vi /etc/snort/database.conf |

1. Restart *service* Snort

|  |
| --- |
| root#/etc/init.d/snort restart |

1. Pada tahap ini Snort sudah dapat difungsikan sebagai *monitoring* IDS, *packet* yang dikategorikan sebagai *event* belum tersimpan ke dalam *database. Service* PostgreSQLperlu dikonfigurasi agar *events* tersimpan ke dalam *database.* Konfigurasi PostgreSQL dapat diubah pada:

|  |
| --- |
| root#vi /etc/postgresql/9.1/main/postgresql.conf |

Ubah konfigurasi seperti berikut:

|  |
| --- |
| .....  listen\_address = postgresql-host-ip  ..... |

listen\_address : diisi dengan alamat IP address dimana *database* terdapat

1. Koneksi TCP/IP dari Snort menuju *database* perlu dikonfigurasi agar Snort dapat berkomunikasi dan mengirimkan data ke *database.*

|  |
| --- |
| root#vi /etc/postgresql/9.1/main/pg\_hba.conf |

Baris berikut ditambahkan pada file konfigurasi diatas:

|  |
| --- |
| host snort snort ip-address-snort-sensor/32 md5 |

ip-address-snort-sensor diisi dengan alamat IP address dari *sensor* Snort.

1. *Service* PostgreSQL perlu di-*restart* agar konfigurasi yang telah diubah dapat berjalan dengan baik:

|  |
| --- |
| root#/etc/init.d/postgres restart |

1. Berikut ini salah satu opsi yang bisa dijalankan pada Snort untuk melakukan *monitoring* jaringan:

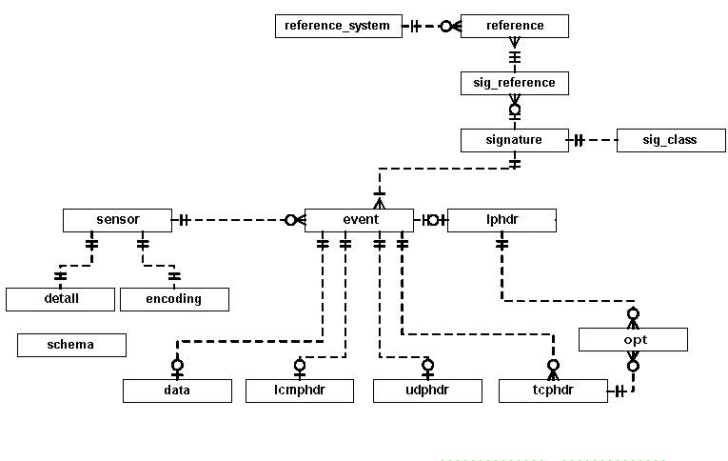
|  |
| --- |
| root#snort -i eth0 -c /ets/snort/snort.conf |

Perintah tersebut akan mengaktifkan *sensor* Snort pada *interface network* eth0 dan menjalankan file konfigurasi snort.conf.

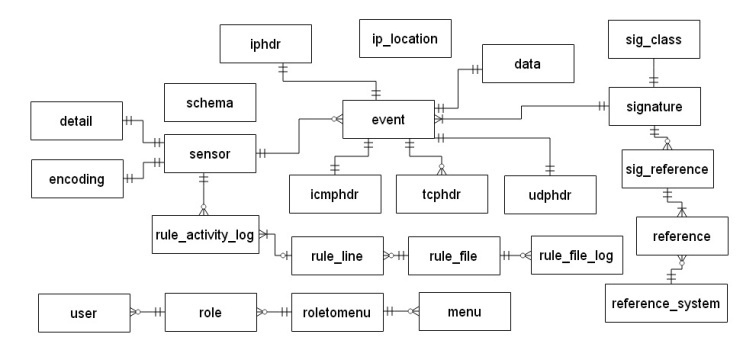
### **Membangun Database IDS Snort**

Tahapan instalasi Snort pada sistem operasi Debian yang telah dijelaskan pada subbab 3.3.1, sepintas juga menjelaskan mengenai cara instalasi *service* PostgreSQL dan *database* standar dari IDS Snort. Pada tahap ini akan dilakukan modifikasi terhadap *database* standar IDS Snort agar sesuai dengan kebutuhan pengembangan aplikasi Mata Garuda dan sesuai dengan kebutuhan administrator.

Berikut ini merupakan *database* standar dari IDS Snort beserta relasi setiap tabel sebelum dilakukan modifikasi:

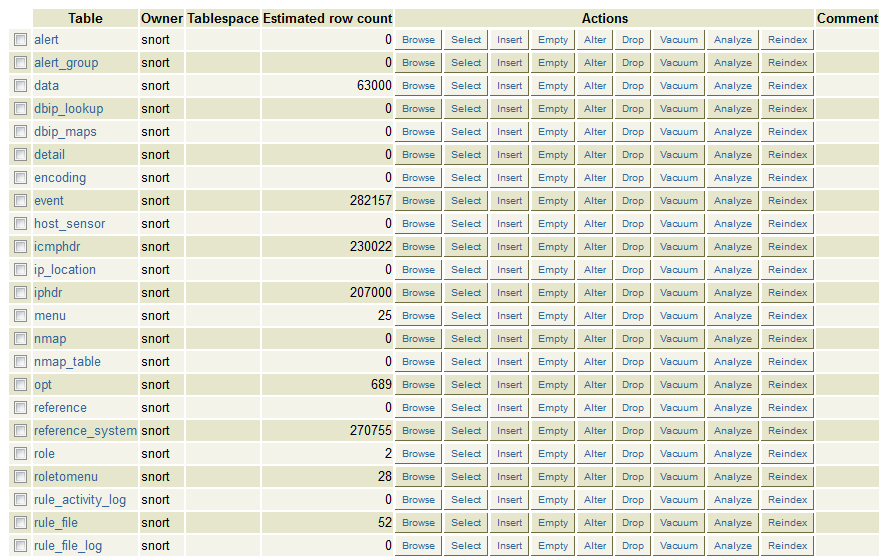


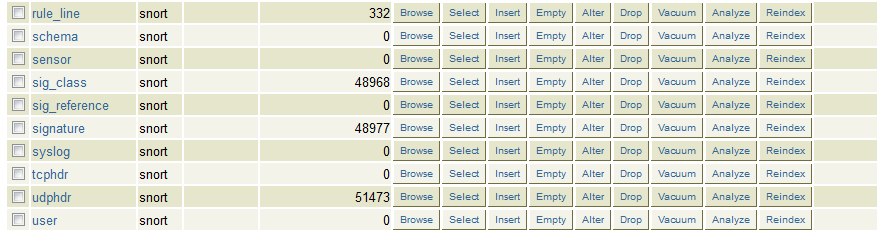
Dari gambar diatas dapat dijelaskan bahwa *database* standar IDS Snort memiliki 15 tabel yang berelasi dan 1 tabel *schema.* Dari *database* tersebut, dimodifikasi dengan menambahkan beberapa tabel untuk kebutuhan aplikasi Mata Garuda seperti yang terlihat pada gambar (mengacu pada rancangan ERD pada subbab 3.1.3):



Gambar diatas memperlihatkan beberapa tabel tambahan:

* 1. user : tabel ini memuat data dan informasi pengguna yang dapat mengakses aplikasi Mata Garuda.
  2. role-roletomenu-menu : relasi tabel ini berisi tentang data menu dan kaitannya dengan pengguna yang mengakses aplikasi, juga menyimpan informasi menu dikarenakan aplikasi Mata Garuda dikembangkan agar menu tampilan bersifat dinamis (bukan *hard-code*).
  3. rule\_activity\_log-rule\_line-rule\_file-rule\_file\_log : relasi tabel ini berisi tentang informasi dan data untuk pengaturan *rules repository* dari aplikasi menuju masing *sensor* Snort.
  4. ip\_location : tabel ini memuat daftar IP *address public* yang ada di dunia beserta masing-masing informasi terkait seperti, *location, ISP, company, latitude, longitude,* dan sebagainya.



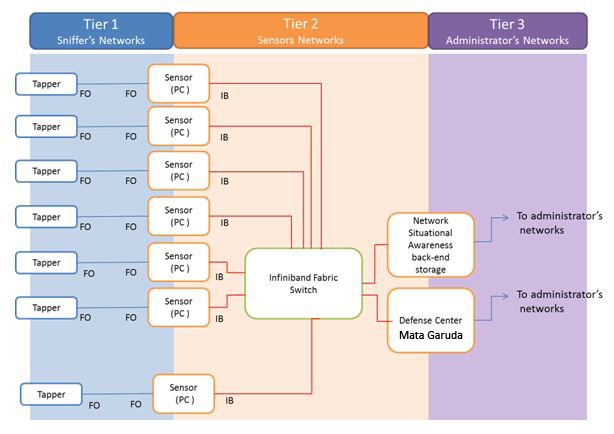


Gambar diatas merupakan tabel-tabel padaaplikasi Mata Garuda yang berasal dari *database* sandar Snort yang telah dimodifikasi. Dapat diakses melalui *web-browser* ataupun *database* GUIyang sesuai.

### **Monitoring Internet (Capture Packet)**

*Intrution Detection System* yang telah dikonfigurasi*,* dalam Proyek Akhir ini yaitu Snort, akan menghasilkan *events* yang nantinya disimpan pada *database* setalah sebelumnya dilakukan *matching* pada *repository rules.*

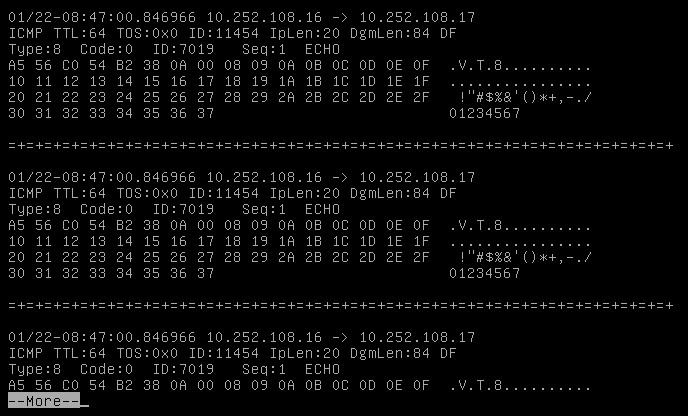
Untuk dapat menghasilkan *events,* Snort akan memantau jaringan. Topologi fisik yang digunakan pada pengembangan Proyek Akhir ini ialah:



Pada gambar diatas dapat dijelaskan bahwa masing-masing *sensor* memiliki paling sedikit dua *interface* jaringan dimana satu *interface* jaringan difungsikan sebagai sarana komunikasi dengan *Defence Center* (server aplikasi Mata Garuda berada) yang berada pada *tier 2,* dan *interface* jaringan lainnya bertindak sebagai *interface sensor,* untuk memantau jaringan internet dan *traffic* yang terjadi didalamnya.

Tiap-tiap *sensor* nantinya akan memantau semua *packet* data yang melalui jaringan yang dipantau dan menghasilkan sebuah *log. Log* biasanya tersimpan pada masing-masing PC *sensor* bukan bersifat *centralized* seperti halnya pada *database.* Perintah berikut untuk memunculkan *event log* dari IDS Snort:

|  |
| --- |
| root#cd /var/log/snort  root#snort -dv -r snort-packet.log |



Gambar diatas menunjukan *log* yang terdapat pada PC *sensor,* dimana data yang disajikan masih berupa *text-based.* Terlihat pada hasil *log* informasi berupa waktu *events, source IP address, destination IP address, protocol type* (ICMP), *ip length,* hingga *data payload* (dalam bentuk *hexadecimal*)untuk masing-masing *events.*

Data tersebut tetap dibutuhkan administrator untuk proses *forensic security network,* namun pada Proyek Akhir ini tidak dibahas lebih lanjut. Data yang dibutuhkan pada Proyek Akhir ini ialah data *event* yang telah tersimpan pada *database* Mata Garuda di *defense center,* dalam artian semua *sensor* yang terhubung dengan Mata Garuda pada *defense center* akan menyimpan data *events* dalam satu *database* utama aplikasi.

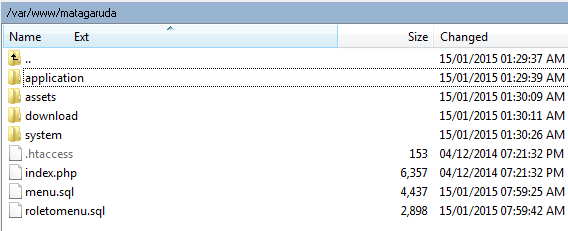


Gambar diatas merupakan hasil *query* dari *database* aplikasi setelah dilakukan *join* beberapa *table,* pada gambar *events* yang dihasilkan berasal dari sensor17 dan memiliki beberapa atribut seperti, *sensor id, event id, hostname, source ip address, destination ip address, timestamp,* dan *data payload.* Untuk *source/destination ip address* data yang ditampilkan masih berupa integer, tentunya akan sulit bagi administrator jaringan untuk menganalisa *events* yang ada sehingga dibutuhkan fungsi untuk mengubah datanya menjadi format standar *ip address.*

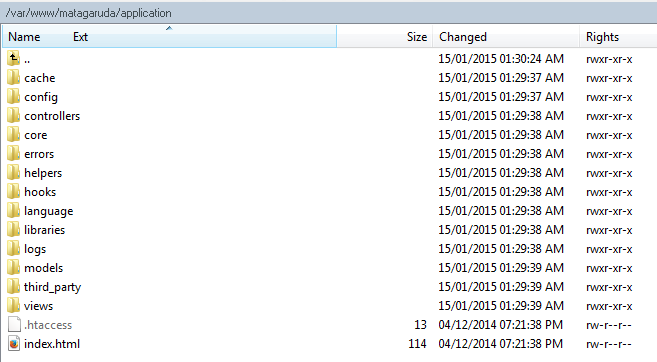
### **Membangun Aplikasi Menggunakan PHP dan CodeIgniter**

Pada tahap ini pembangunan aplikasi antar muka IDS Snort Mata Garuda akan dijelaskan berdasarkan *use case* yang telah dipaparkan pada subbab 3.1.3. Bahasa pemrograman PHP dan *framework* CodeIgniter menjadi *tools* utama yang digunakan pada tahapan ini.

CodeIgniter menerapkan fungsi MVC yaitu *Model, View,* dan *Controller.* Semua fungsi *controller* untuk menyajikan data seperti *dashboard, events, event statistic, rule,* dan *setting* berada pada direktori *controller,* komunikasi aplikasi dengat tabel-tabel terkait berada pada direktori *model,* dan tampilan dari aplikasi Mata Garuda berbasis *web* ada pada direktori *view.* Pada Proyek Akhir ini aplikasi dibangun diatas sistem operasi Debian 7 sehingga direktori dari CodeIgniter yang akan dikembangkan menjadi aplikasi Mata Garuda ditaruh pada *web directory* Debian “/var/www/”



Dari gambar diatas terlihat direktori CodeIgniter yang telah diubah namanya menjadi **matagaruda** (disesuaikan dengan kebutuhan pengembangan) berada dalam direktori “/var/www/” pada sistem operasi Debian 7.



Gambar diatas menunjukan struktur direktori *application,* pada direktori ini terdapat direktori *controllers, models,* dan *views* sebagai direktori utama tempat dimana modul-modul yang dirancang akan dikembangkan. Jika ingin melakukan pengembangan tidak harus mengubah keseluruhan sistem, cukup menambahkan file fungsi pada direktori *controller* yang melakukan pengolahan dan *scripting* untuk menampilkan data, menambahkan file untuk tampilan yang berada pada direktori *views,* dan memanfaatkan direktori *models* jika modul yang dikembangkan mengambil data dari *database.* Hal tersebut merupakan salah satu keunggulan konsep *modularity* dimana dalam hal mengembangkan modul dapat dipisah dari sistem utama dan dikerjakan secara mandiri terlebih dahulu, lalu pada akhirnya dilakuka penggabungan dengan sistem utama.

CodeIgniter perlu dihubungkan dengan *database* yang telah dibuat sebelumnya (subbab 3.3.2) agar aplikasi dapat mengambil data dari *database* dan menyajikannya kepada administrator.



Gambar diatas merupakan konfigurasi file “/var/www/matagaruda/application/config/database.php”, untuk menghubungkan aplikasi Mata Garuda dengan database.

***hostname***: diisi dengan *hostname* dimana *database* berada

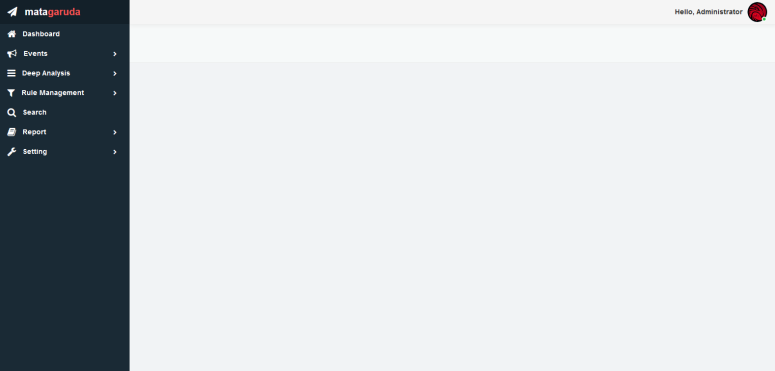
***username*** : diisi dengan username *database* IDS yaitu *snort*

***password*** : merupakan *password* yang digunakan agar dapat mengakses *database*

***database*** : diisi dengan nama *database* yang telah dibangun sebelumnya yaitu *snort*

***dbdriver*** : diisi dengan postgres karena DBMSyang digunakan dalam Proyek Akhir ini adalah PostgreSQL

Sebelum melakukan pengembangan terhadap fungsi-fungsi atau modul-modul yang telah dirancang sebelumnya, membuat suatu *layout* tampilan kosong dengan menu-menu yang disesuaikan dari *database* akan sangat membantu sehingga pengembangan aplikasi per-modul kedepannya akan semakin mudah.



Gambar diatas merupakan *layout* dari aplikasi Mata Garuda dan disertakan juga menu-menu aplikasi pada bagian navigasi sebelah kiri. Ruang kosong pada sisi kanan merupakan ruang bagi tiap-tiap modul untuk menyajikan data dan informasi. File *layout* diatas disimpan dengan nama *container.php* dan berada di dalam direktori *views* pada CodeIgniter. Nantinya modul-modul yang akan dikembangkan bisa menggunakan *layout* ini dengan memanggil *container.php* pada baris *coding controller* tiap modul*.*

1. ***Dashboard***

*Dashboard* yang bertindak sebagai pembuka aplikasi yang menyajikan data *summary* kepada administrator, walaupun hanya data *summary* namun diharapkan data yang disajikan akan memberikan setidaknya setengah dari informasi yang dapat disajikan oleh keseluruhan aplikasi Mata Garuda.

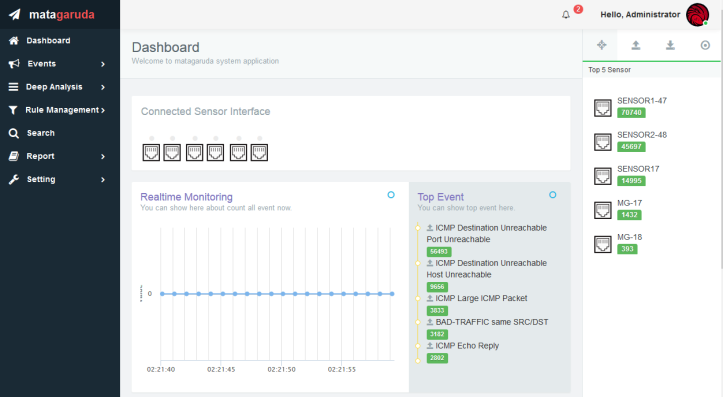
Data atau informasi yang ditampilkan pada *dashboard* mencakup:

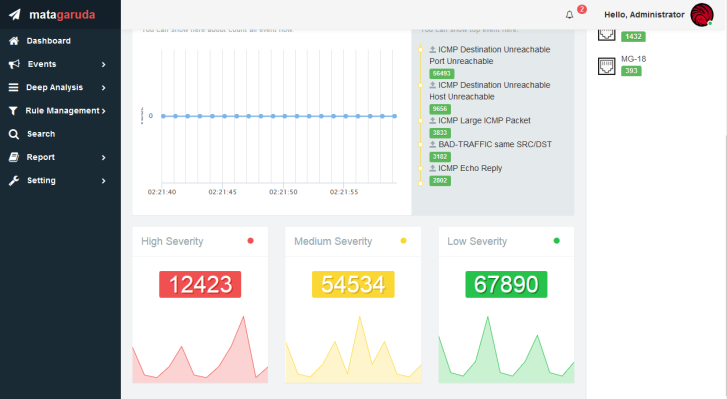
1. Informasi *Sensor*
2. Grafik *Realtime* *Monitoring*
3. *Top* 5 *Signature*
4. *Top 5 Sensor*
5. *Top 5 Source/Destination IP address*
6. *Top 5 Destination Port TCP/UDP*
7. *Summary Severity*

|  |  |  |
| --- | --- | --- |
| Fungsi | Tabel Relasi | Field |
| Informasi Sensor | sensor | hostname, ip\_address |
| Grafik *Realtime Monitoring* | - | - |
| *Top 5 Signature* | event, signature | sig\_name |
| *Top 5 Sensor* | event, sensor | hostname |
| *Top 5 Source/Destination IP Address* | event, iphdr | ip\_src, ip\_dst |
| *Top 5 Destination Port TCP/UDP* | event, tcphdr, udphdr | tcp\_dport, udp\_dport |
| *Summary Severity* | event, signature | sig\_priority |

Tabel diatas menunjukan tabel apa saja pada *database* beserta *field* yang sesuai yang digunakan oleh setiap fungsi modul untuk menyajikan data dan informasi. Untuk grafik *Realtime Monitoring,* datanya tidak berasal dari *database* namun menggunakan fungsi PHP untuk menciptakan *file* JSON yang berisi data dan informasi *event* yang juga berasal dari IDS Snort.

|  |  |  |
| --- | --- | --- |
| ***Controllers*** | : | *dashboard.php* |
| ***Models*** | : | *dashboard\_event\_model.php*  *dashboard\_sensor\_model.php* |
| ***Views*** | : | *dashboard.php* |





Gambar diatas merupakan tampilan dari *dashboard* aplikasi Mata Garuda yang telah memuat berbagai Informasi *Summary.*

1. ***Events***

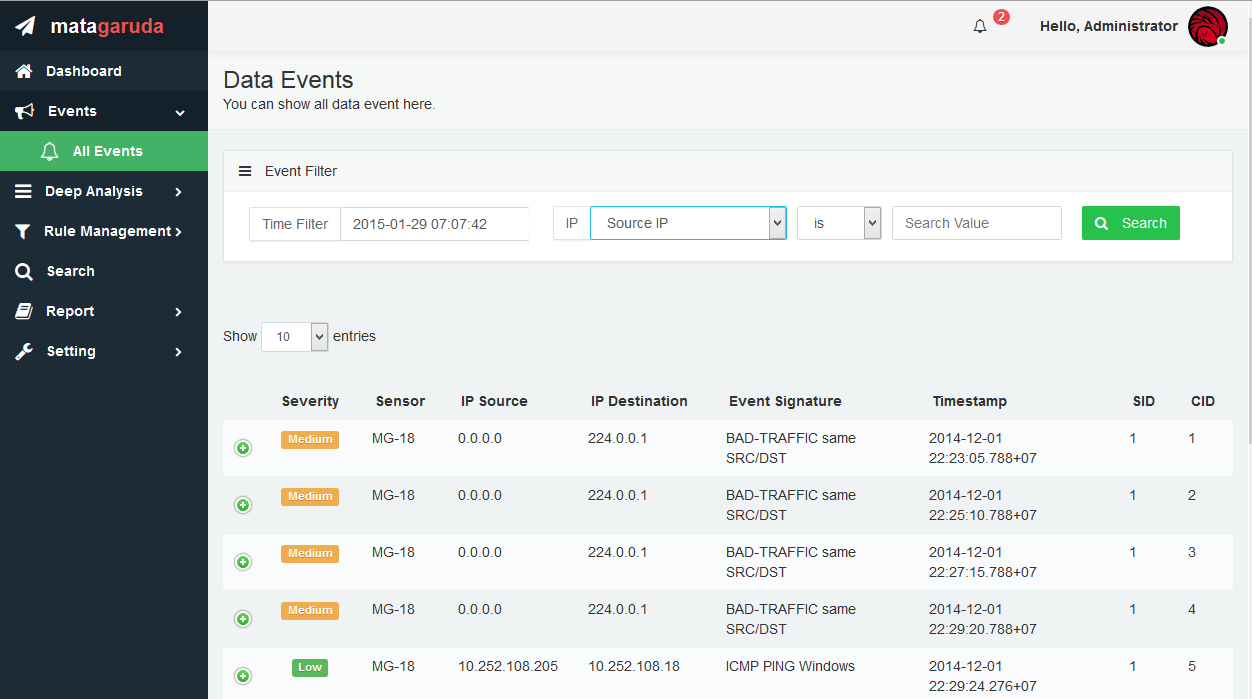
Menampilkan keseluruhan *record event* dari masing-masing *sensor* dan ditampilkan pada satu tampilan tabel informatif. Informasi dan aksi yang ditampilkan pada halaman *event* yaitu:

1. *Record Event* semua *sensor*
2. *Filter Event*

|  |  |  |
| --- | --- | --- |
| Fungsi | Tabel Relasi | Field |
| *Record event* | event, sensor, signature, iphdr, icmphdr, tcphdr, udphdr | sig\_priority, hostname, ip\_src, ip\_dst, sig\_name, timestamp, sid, cid |

Tabel diatas menunjukan relasi tabel pada *database* beserta *field* yang digunakan oleh modul *event* untuk menyajikan data.

|  |  |  |
| --- | --- | --- |
| ***Controllers*** | : | *event.php* |
| ***Models*** | : | *event\_model.php* |
| ***Views*** | : | *event\_data.php, event\_model.php* |



Gambar diatas merupakan tampilan dari *event* Mata Garuda yang menyajikan informasi dari masing-masing *event sensor.*

*Event* dapat di-*filter* berdasarkan waktu kejadian, kategori *event* (pada contoh diatas berdasasrkan *source IP address*)untuk mendapatkan hasil sesuai kebutuhan analisa administrator jaringan.

1. **Analisa Packet/Data**

Pada modul ini menampilkan data *event* dalam bentuk grafik garis dan *pie chart.* Grafik dimaksudkan agar administrator dapat mengetahui banyak/jumlah *event* yang terjadi persatuan waktu dan mendapatkan kesimpulan mengenai *event* yang terjadi pada jaringan internet.

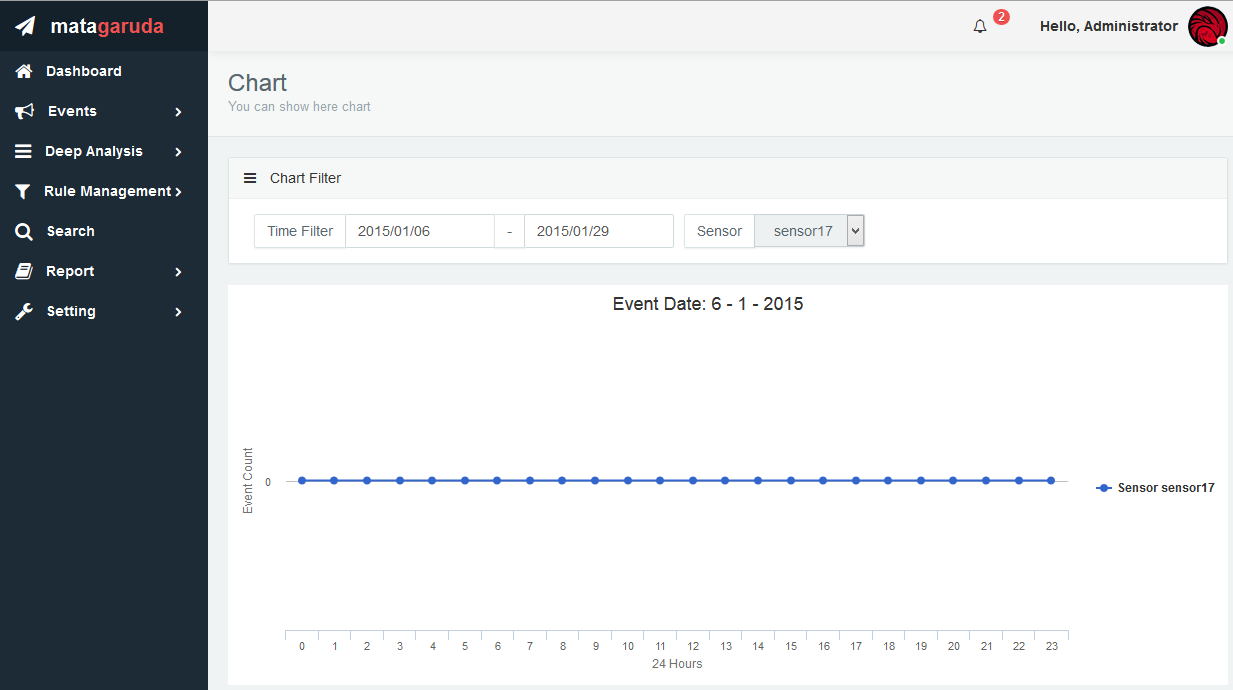
Informasi dan aksi yang ditampilkan pada halaman *Deep Analysis* untuk submenu *Event Statistic* ialah:

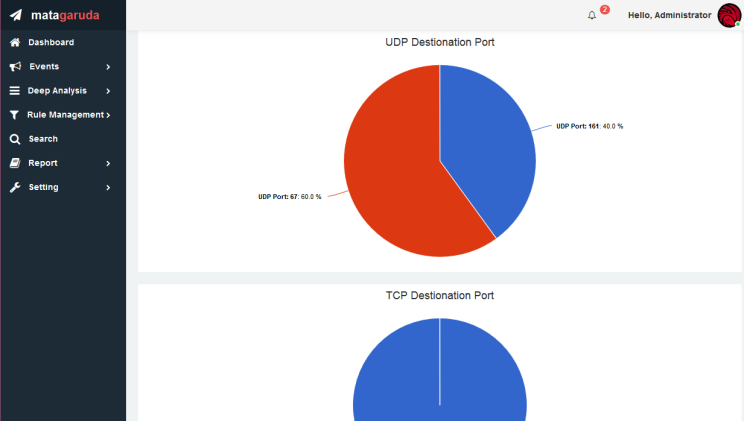
1. Grafik garis *event* per satuan waktu
2. Jumlah serangan pada *protocol* dalam bentuk *pie chart*

|  |  |  |
| --- | --- | --- |
| Fungsi | Tabel Relasi | Field |
| Grafik garis | event, sensor | *count(event),* *hostname* |
| Pie Chart | *event, sensor, tcphdr, udphdr* | *count(event), hostname, tcp\_dport, udp\_dport* |

Tabel diatas menunjukan nama fungsi yang ditampilkan pada modul *Event Statistic* beserta relasi tabel yang digunakan pada *database* aplikasi.

|  |  |  |
| --- | --- | --- |
| ***Controllers*** | : | *chart.php* |
| ***Models*** | : | *chart\_model.php* |
| ***Views*** | : | *chart\_view.php* |

****

****

Gambar diatas menunjukan grafik garis dan *pie chart* dari aplikasi Mata Garuda yang memberikan informasi jumlah *event* per satuan waktu dan juga jumlah serangan ke *port* tertentu pada masing-masing *protocol* TCP dan UDP. Dari gambar diatas juga dapat dilihat bahwa grafik dapat di-*filter* sesuai dengan masukkan waktu dari administrator untuk mendapatkan hasil yang berbeda, pun demikian untuk nama *sensor.*

1. ***Rule***

Modul *Rule* tidak menjadi kosentrasi atau pengembangan dari penulis dalam mengerjakan Proyek Akhir ini, dimana modul ini dikembangkan oleh tim pengembang lainnya (penulis hanya mengembangkan *core system* Mata Garuda) namun dirasa perlu untuk dipaparkan agar salah satu fungsi dan keunggulan dari Mata Garuda bisa terlihat.

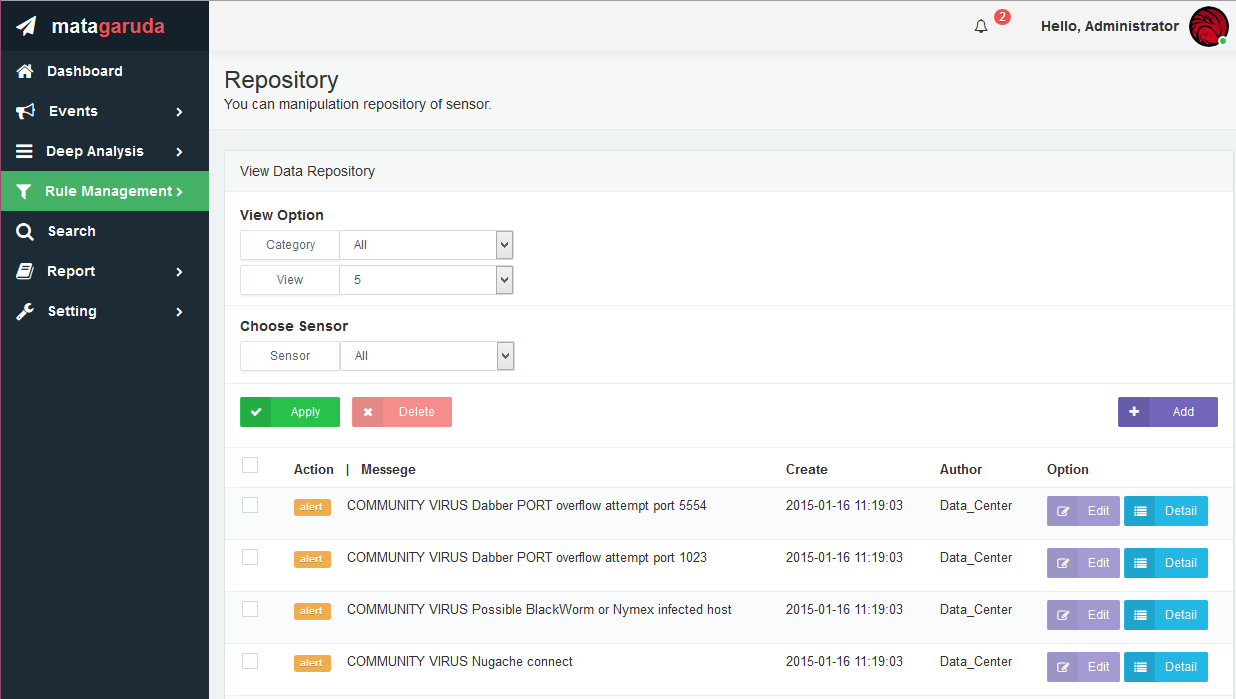
Informasi dan aksi yang dapat ditampilkan yaitu:

1. *Repository Rules*
2. *Management Rules*

|  |  |  |
| --- | --- | --- |
| Fungsi | Tabel Relasi | Field |
| *Repository* dan *Management Rules* | *sensor, rule\_activity\_log, rule\_file, rule\_file\_log, rule\_line* | *hostname, action, message, create, author* |

Tabel diatas merupakan tabel relasi pada *database* aplikasi yang digunakan oleh modul *Rules* untuk menyajikan data pada aplikasi Mata Garuda.

|  |  |  |
| --- | --- | --- |
| ***Controllers*** | : | *rules.php* |
| ***Models*** | : | *rules\_model.php* |
| ***Views*** | : | *rules.php* |



Gambar diatas menunjukan tampilan dari modul *rules* berikut daftar *rules* yang ada (*repository*) beserta aksi yang dapat dilakukan (*management*).

1. ***Report***

Pada menu *report,* menampilkan data transaksional yang berguna untuk pelaporan hasi analisa *monitoring* jaringan internet. Informasi yang dapat dilihat berupa:

1. Data Transaksi/*Report* per Hari
2. Data Transaksi/*Report* per Bulan
3. Data Transaksi/*Report* per Tahun

|  |  |  |
| --- | --- | --- |
| Fungsi | Tabel Relasi | Field |
| *Transaksi/Report* | *event, sensor, signature, iphdr, icmphdr, tcphdr, udphdr* | *hostname, signature, ip\_src, ip\_dst, sig\_name, icmp\_type, tcp\_dport, udp\_dport* |

Tabel diatas merupakan tabel relasi yang digunakan oleh modul *report* untuk menyajikan data transaksional/*report.*

|  |  |  |
| --- | --- | --- |
| ***Controllers*** | : | *report.php* |
| ***Models*** | : | *report\_model.php* |
| ***Views*** | : | *report\_view.php* |

1. **Konfigurasi Aplikasi**

Modul konfigurasi hanya melakukan pengaturan pada aplikasi Mata Garuda dan tidak melakukan pengubahan konfigurasi pada sistem IDS. Seperti yang telah dijelaskan pada Alur Aplikasi (subbab 3.1 : Deskripsi Umum Sistem).

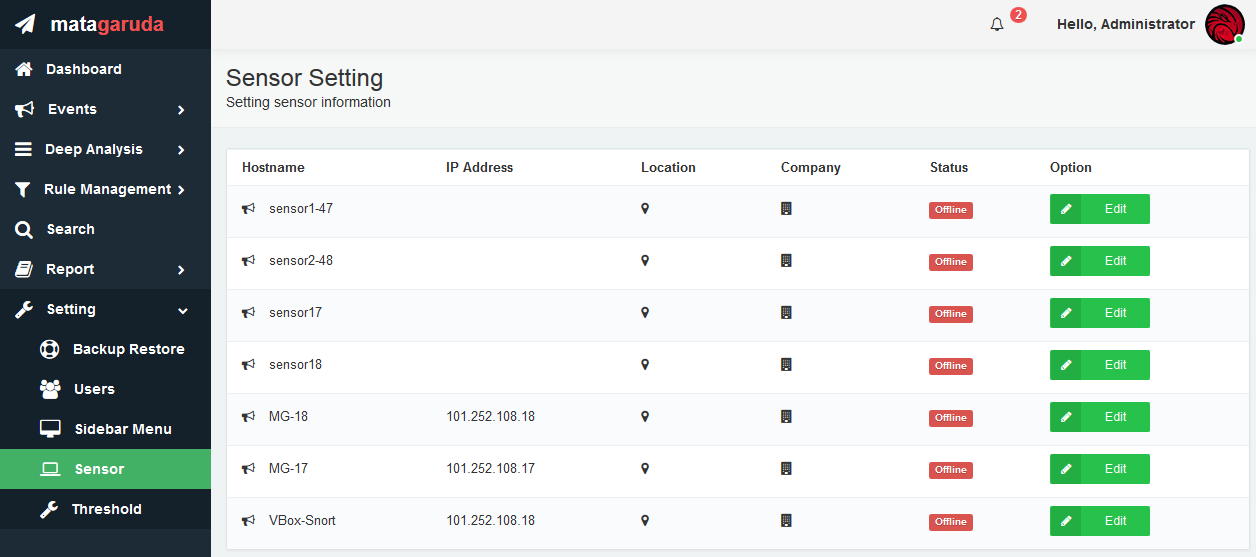
Informasi dan aksi yang dapat dilakukan pada menu *setting* (Konfigurasi Aplikasi) yaitu:

1. Konfigurasi *sensor* pada *database*
2. Konfigurasi *threshold*

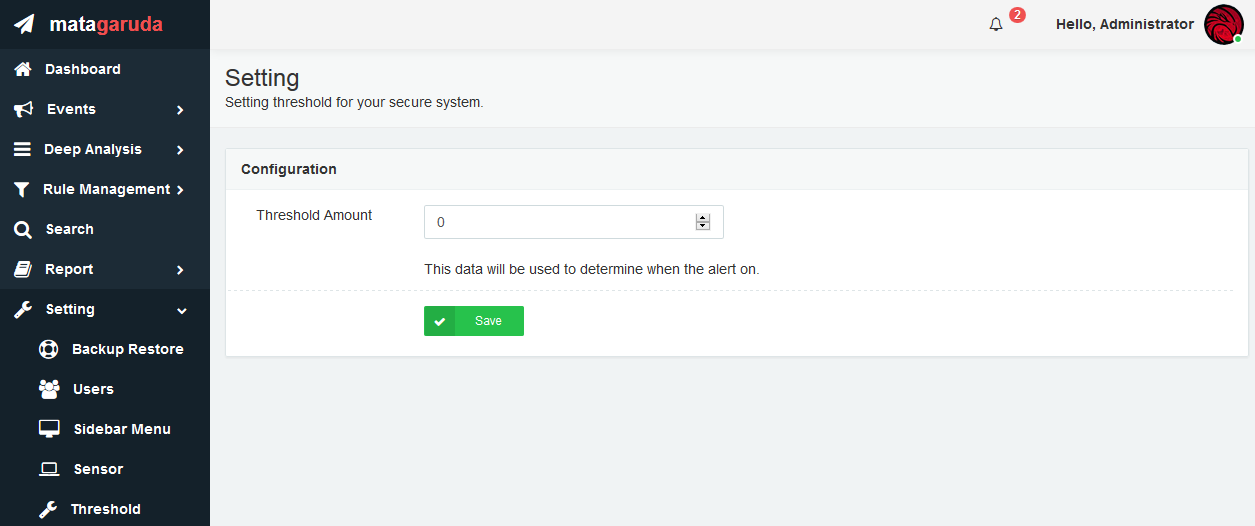
|  |  |  |
| --- | --- | --- |
| Fungsi | Tabel Relasi | Field |
| Konfigurasi *sensor* | *sensor* | *hostname, ip address, location, company* |

Tabel diatas hanya menunjukan tabel relasi yang digunakan oleh fungsi Konfigurasi *sensor,* sementara fungsi konfigurasi *threshold* tidak tersimpan pada *database.*

|  |  |  |
| --- | --- | --- |
| ***Controllers*** | : | *setting.php* |
| ***Models*** | : | *setting\_model.php* |
| ***Views*** | : | *setting\_sensor.php, setting\_sensor\_modify.php setting\_threshold.php* |



Gambar diatas menunjukan tampilan dari *setting-*an *sensor* yang menyajikan informasi *hostname, ip address sensor, location, company,* dan status koneksi. Pada gambar diatas status koneksi masih *offline* dikarenakan IP untuk masing-masing *sensor* belum dikonfigurasi.



Gambar diatas menunjukan tampilan dari *setting-*an *threshold,* dimana *threshold* akan digunakan oleh apliksasi sebagai batasan *realtime monitoring* pada menu *dashboard.*

# **BAB IV**

# **PENGUJIAN SISTEM DAN APLIKASI**

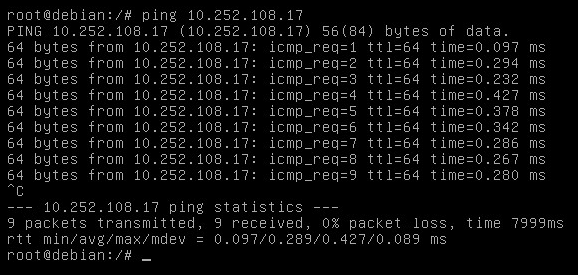
Pada bab ini dibahas mengenai hasil pengujian aplikasi Mata Garuda. Pengujian dilakukan untuk mendapatkan hasil apakah aplikasi Mata Garuda ini (pengembangan *core system*) sudah sesuai dengan desain dan kebutuhan yang telah dirancang.

## **Pengujian Terhadap IDS Snort**

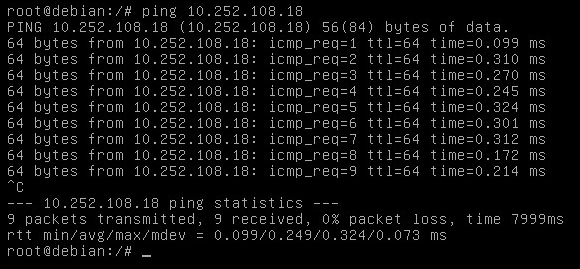
Pengujian dilakukan untuk mengetahui apakah IDS Snort sudah berjalan dengan baik atau belum

### **Pengujian Konektifitas**

Pengujian pertama yaitu pada konektifitas dari *sensor* IDS Snort. Tahap ini akan melakukan pengujian apakah masing-masing *sensor* IDS sudah terhubung ke *defense center* (dalam hal ini *server* Aplikasi Mata Garuda) atau belum. Pada pengujian kali ini menggunakan dua *sensor*:



Gambar diatas menunjukan hasil koneksi dari *defense center* dengan alamat IP *address* 10.252.108.16 menuju *sensor* 1 dengan alamat IP *address* 10.252.108.17 dan hasilnya ialah *reply* yang menandakan bahwa *sensor* 1 dapat berkomunikasi dengan *defense center.*



Gambar diatas menunjukan hasil koneksi dari *defense center* dengan alamat IP *address* 10.252.108.16 menuju *sensor* 2 dengan alamat IP *address* 10.252.108.18 dan hasilnya ialah *reply* yang menandakan bahwa *sensor* 2 dapat berkomunikasi dengan *defense center.*

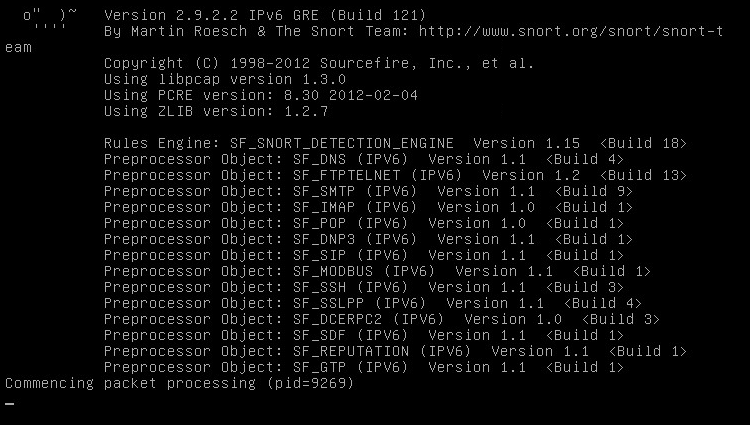
### **Pengujian *Running* Snort**

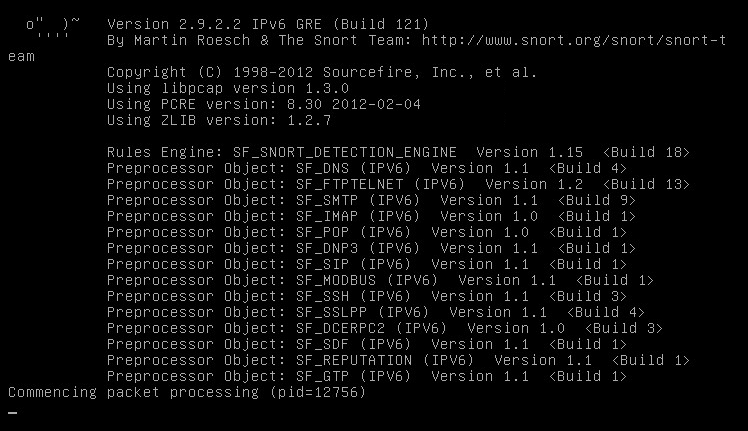
Setelah konektifitas *sensor* diuji, pengujian selanjutnya yaitu pada *service* Snort masing-masing *sensor.* Snort pada tiap-tiap *sensor* diasumsikan sudah dikonfigurasi dengan baik (mengacu pada subbab 3.3.1: Instalasi dan Konfigurasi Snort), dibawah ini merupakan konfigurasi penting pada Snort:

running17-1.PNG

running18-1.PNG

Dapat diperhatikan bahwa untuk masing-masing *sensor* memiliki nama yang berbeda beda, pada pengujian kali ini untuk *sensor* 1 (10.252.108.17) memiliki nama **sensor17** dan untuk *sensor* 2 (10.252.108.18) memiliki nama **sensor18**.





Gambar diatas menunjukkan bahwa IDS Snort pada *sensor* 1 (*sensor17,* 10.252.108.17) sudah dapat berjalan dengan baik dengan cara menjalankan perintah:

|  |
| --- |
| root#snort -c /etc/snort/snor.conf |

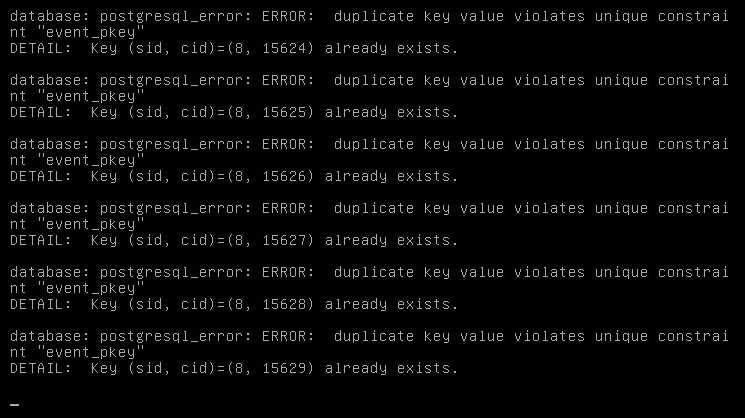
Untuk daftar lengkap perintah menjalankan IDS Snort pada sistem (disesuaikan dengan kebutuhan) dapat mengakses alamat lengkap Snort (<http://snrt.org>)

### **Pengujian *Catpure Events***

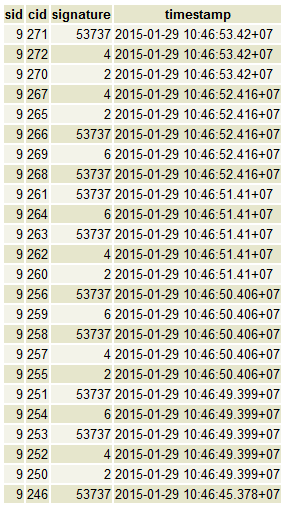
Bagian terpenting untuk sebuah IDS yaitu apakah IDS tersebut mampu melakukan *capture packet* pada jaringan yang di-*monitoring* dan menyimpannya pada *database* dan *log* jika dikategorikan sebagai sebuah *event.*

Berikut merupakan hasil pengujian IDS Snort dalam melakukan *capture events,* pada pengujian ini Snort dalam keadaan *running* dan dilakukan serangan dari jaringan yang dipantau oleh masing-masing *sensor.* Menggunakan serangan sederhana yaitu melakukan *ping* ke masing-masing *sensor* dengan *size* 900 *bytes.*

|  |
| --- |
| windows C:> ping 10.252.108.17 –t –l 900 |
| windows C:> ping 10.252.108.18 –t –l 900 |



Gambar diatas menunjukan proses *capture* yang dilakukan oleh *sensor17* dan langsung dimasukkan ke dalam *database* yang terdapat pada *defense center* aplikasi Mata Garuda. Pada pengujian tersebut terdapat *error* namun bukan karena IDS Snort tidak dapat meng-*capture packet* melainkan dikarenakan *redudancy* data yang terdapat pada *database* aplikasi Mata Garuda.

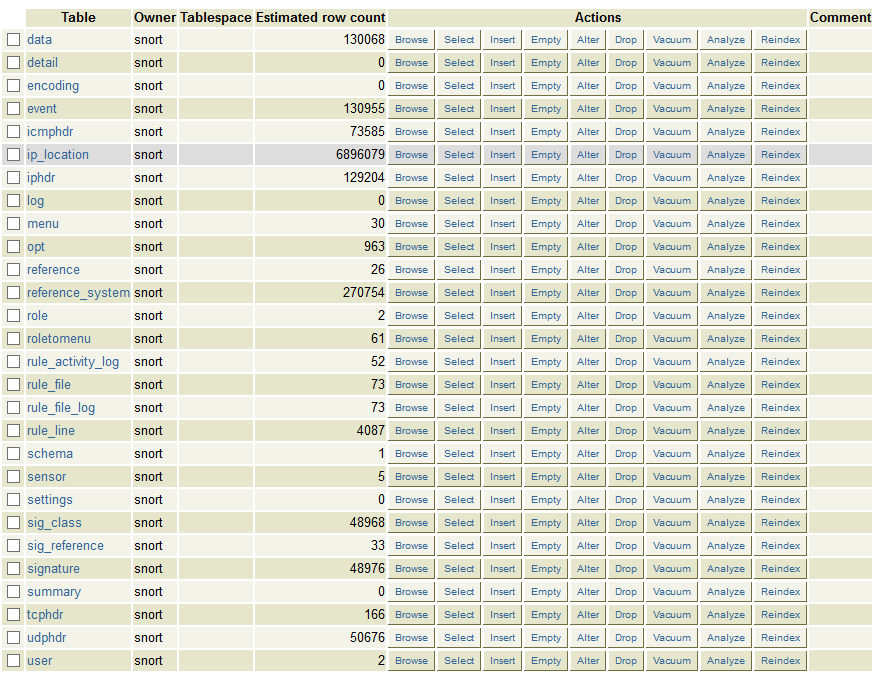


Gambar diatas merupakan hasil *capture sensor18,* yang langsung ditampilkan pada *database* aplikasi Mata Garuda (bukan melalui *console* seperti hasil *capture sensor17*).

## **Pengujian Database**

*Database* yang telah dibangun dan dirancang perlu dilakukan pengujian untuk mengetahui apakah *database* tersebut mampu menyimpan hasil *capture* IDS Snort atau tidak.

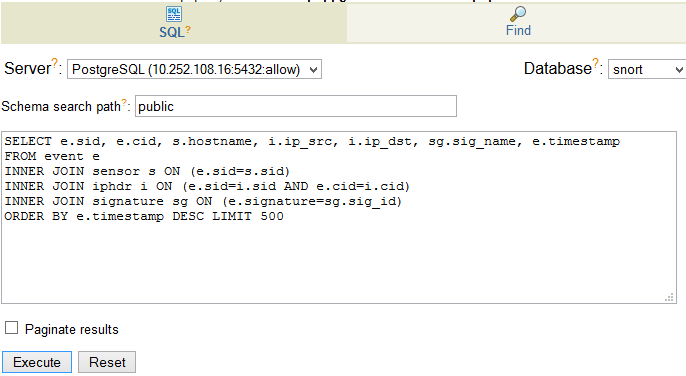
### **List Table pada Database**



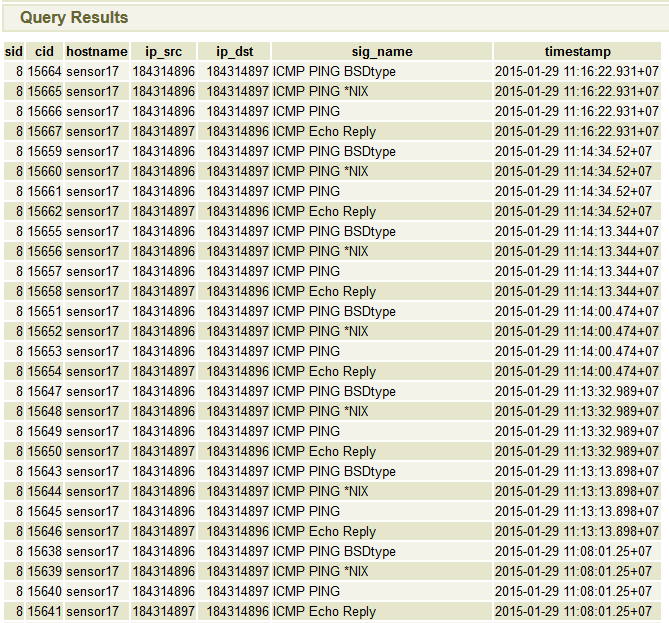
Gambar diatas merupakan *list* tabel yang terdapat pada *database* aplikasi Mata Garuda, terlihat untuk beberapa tabel sudah terisi dengan beberapa *record* (*estimated row count*)

### **Pengujian Query**

Dari hasil pengujian subbab 4.1: Pengujian Terhadap IDS Snort, sepintas telah disajikan data hasil *capture* dari *database* yang membuktikan bahwa *database* aplikasi Mata Garudasudah mampu menyimpan data hasil *capture* tiap *sensor,* namununtuk memastikan lebih lanjut perlu dilakukan pengujian lagi untuk *query database.*



Gambar diatas merupakan salah satu contoh *query* untuk menampilkan data dan informasi sesuai kebutuhan administrator.



Gambar diatas merupakan hasil dari *query* yang telah dijelaskan sebelumnya. Dari dua hasil pengujian terhadap *database* dapat disimpulkan bahwa *database* aplikasi Mata Garuda sudah dapat berjalan dengan baik dan mampu menyimpan data dari IDS Snort serta mampu menyajikannya dari hasil *query* buatan administrator.

## **Pengujian Aplikasi Mata Garuda**

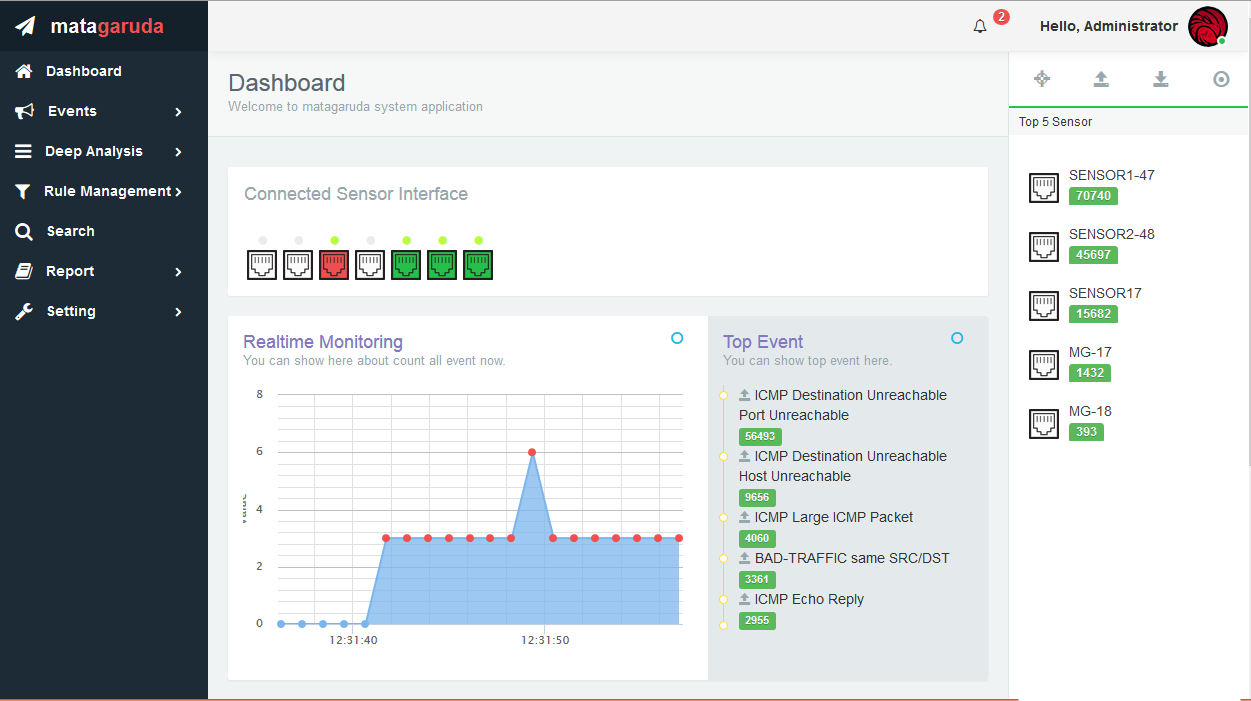
Pada tahap ini akan melakukan pengujian terhadap antar muka aplikasi Mata Garuda, dengan menjalankan beberapa masukkan dari administrator.

### **Pengujian Menu *Dashboard***

Pada tahap ini akan dilakukan pengujian terhadap menu *dashboard,* mulai dari pengujian *realtime monitoring* hingga memunculkan data *severity.*

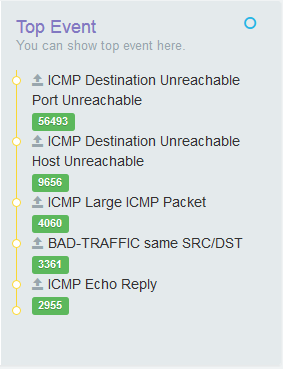
1. *Real Time Monitoring*

Pengujian untuk *realtime monitoring* dilakukan dengan menyerang *sensor* IDS menggunakan *ping of death.*

**

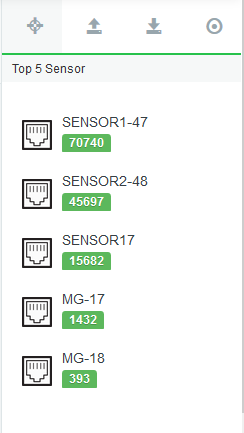
Dari gambar diatas terlihat bahwa grafik berubah sesuai dengan waktu dan banyaknya serangan yang terjadi.

1. *Top Signature*



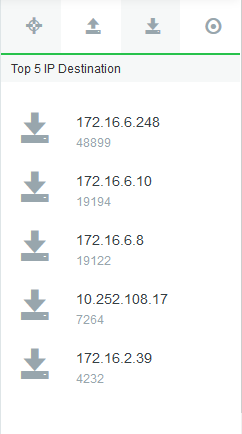
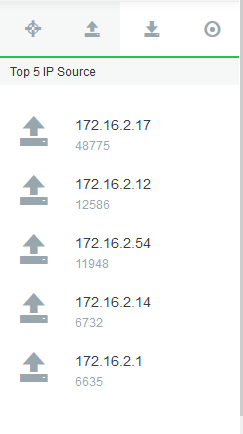
Gambar diatas menunjukkan 5 jenis *signature* dengan jumlah yang paling banyak dari semua *sensor* yang ada.

1. *Top 5 Sensor*



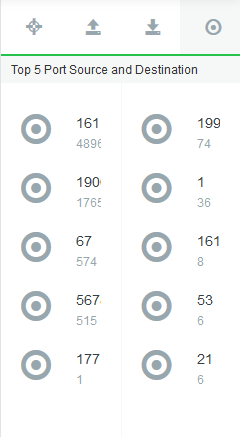
Gambar diatas merupakan tampilan yang menyajikan banyaknya jumlah *event* untuk masing-masing *sensor,* dimana pada tampilan diatas menyajikan 5 *sensor* dengan jumlah *event* paling banyak.

1. *Top 5 Source/Destination IP Address*



Gambar diatas menampilkan 5 *source* IP *address* dan *destination* IP *address* berdasarkan jumlah *events.*

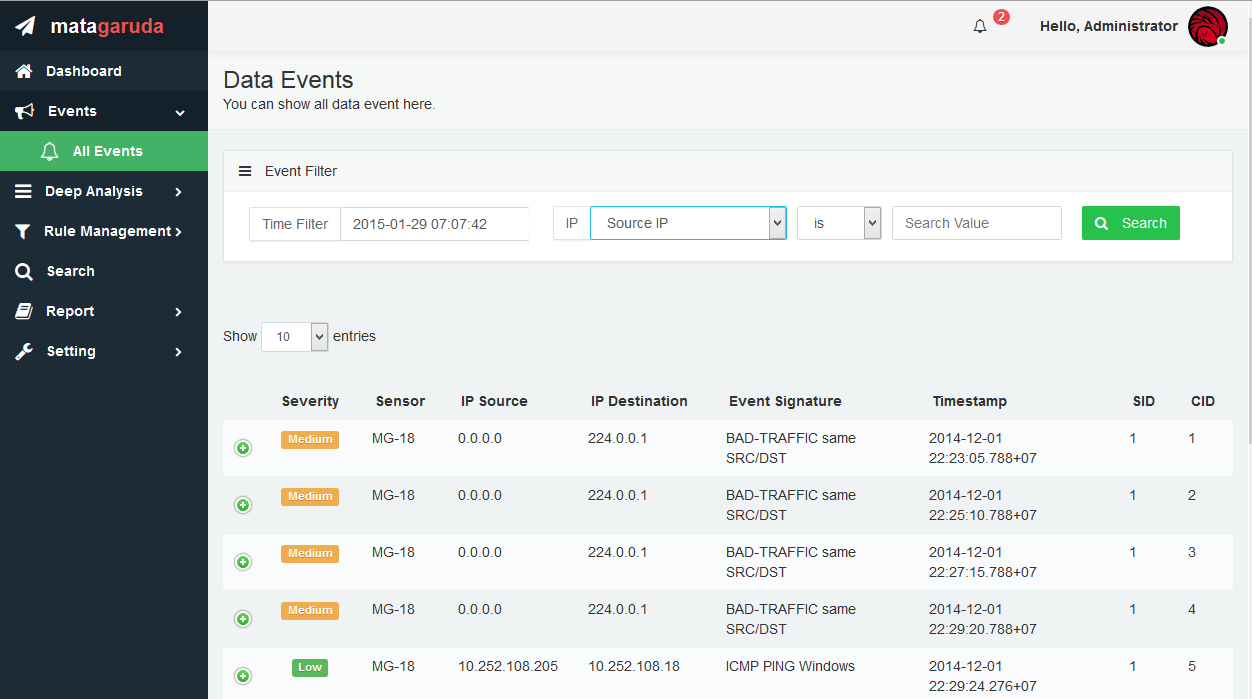
1. *Top 5 Port TCP/UDP*

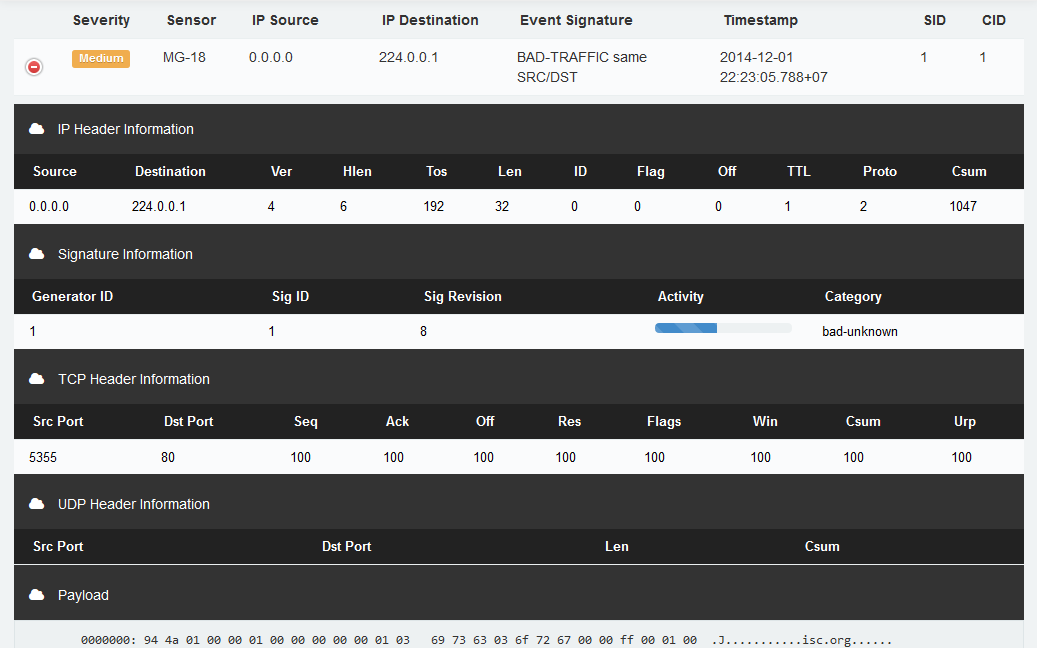


Dari gambar diatas terlihat jenis *port* dan jumlah serangan atau *events* pada *port* tersebut.

### **Pengujian Menu *Events***

Pada tahap pengujian ini akan menampilkan hasil dari *event* yang berhasil tersimpan ke dalam *database* aplikasi oleh masing-masing *sensor* IDS Snort.

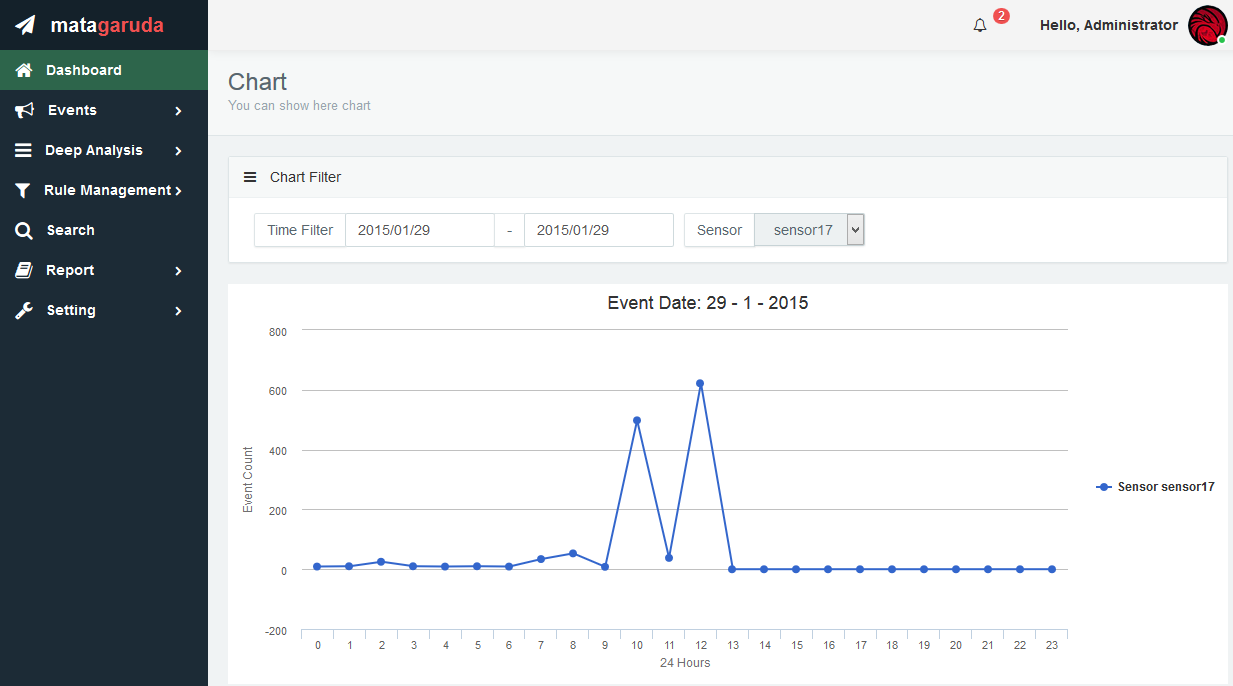




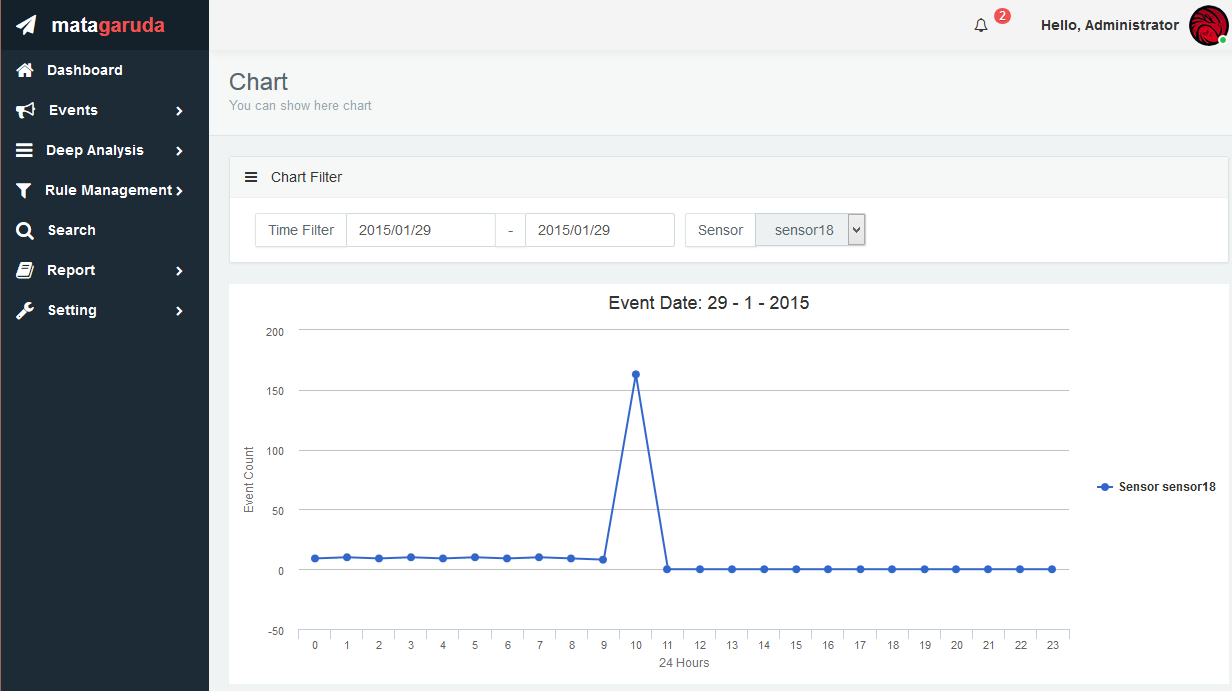
Pada gambar diatas terlihat tampilan *list* dari *event* yang berhasil disimpan ke *database,* dan juga detail untuk masing-masing *event* yang menyajikan lebih banyak lagi informasi suatu *event.* Untuk detail *event* bersifat *hidden* dan membutuhkan aksi dari administrator untuk menampilkan data detailnya.

### **Pengujian Menu *Event Statistic***

*Event statistic* pada pengujian kali ini ditampilkan dalam bentuk grafik garis dan *pie chart.*

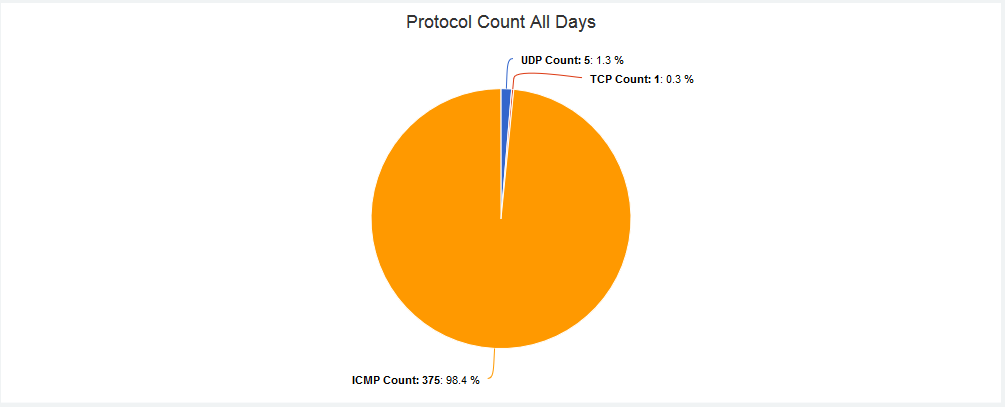
**

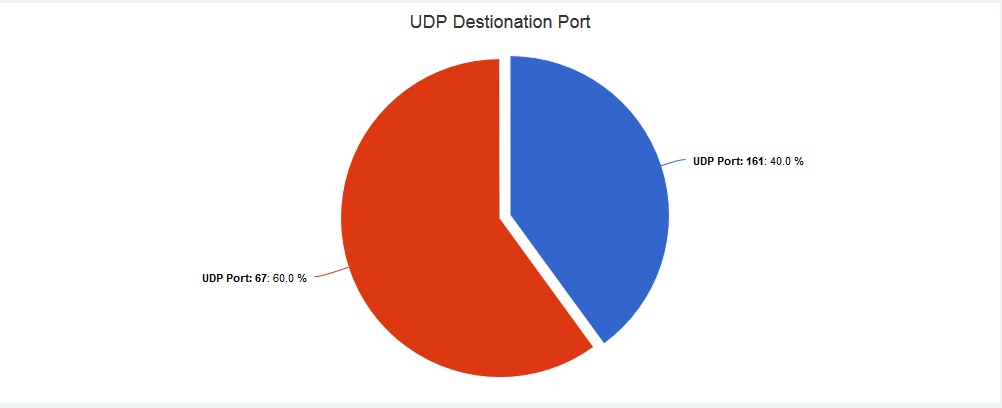
Pada gambar diatas terlihat grafik garis untuk *sensor17* pada tanggal 29-01-2015 yang menampilkan jumlah *events* yang berbeda-beda setiap jamnya. Hasil yang berbeda ditunjukan pada grafik garis untuk *sensor18* pada tanggal pengujian yang sama, seperti berikut:

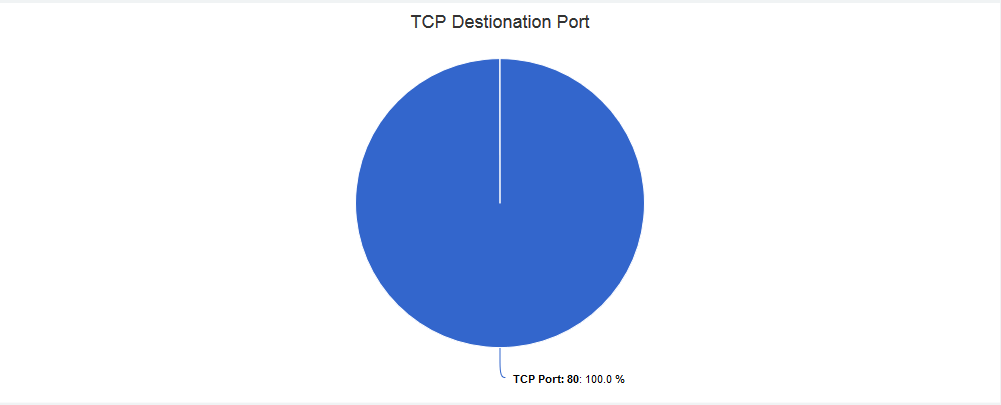


Gambar diatas merupakan tampilam grafik garis untuk *sensor18.*

Selain menguji grafik garis, pada halaman yang sama juga dilakukan pengujian untuk *pie chart protocol* pada *event* di dalam *database.*



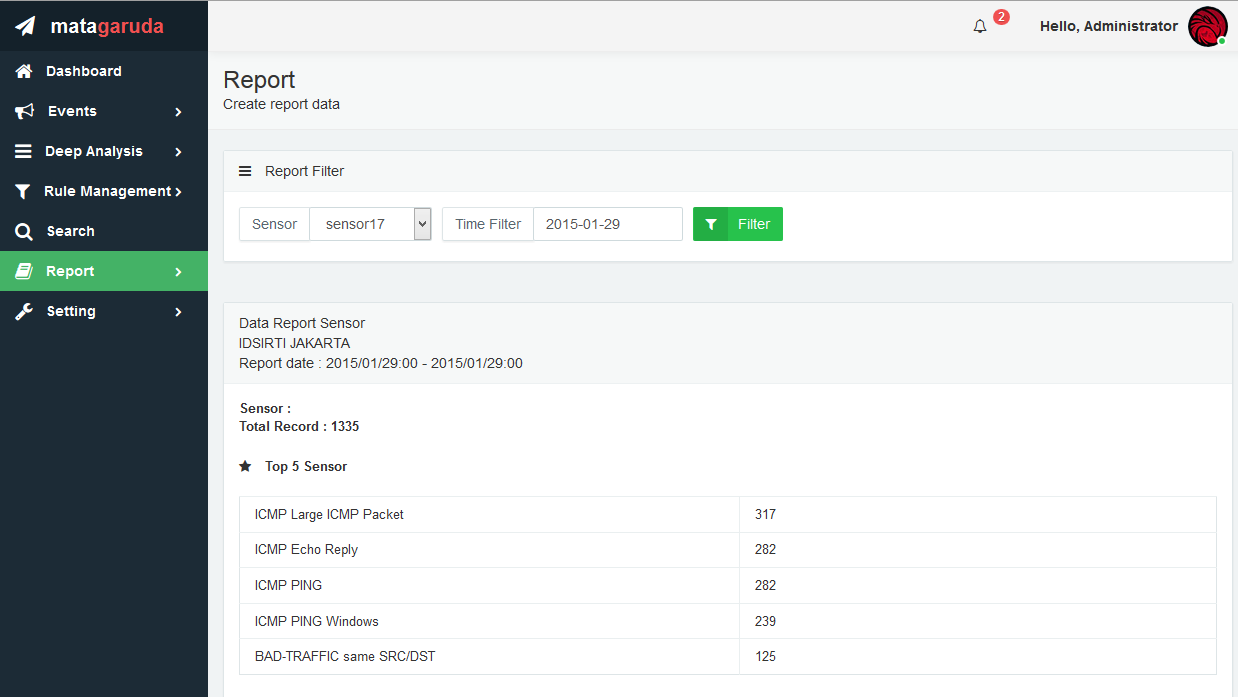


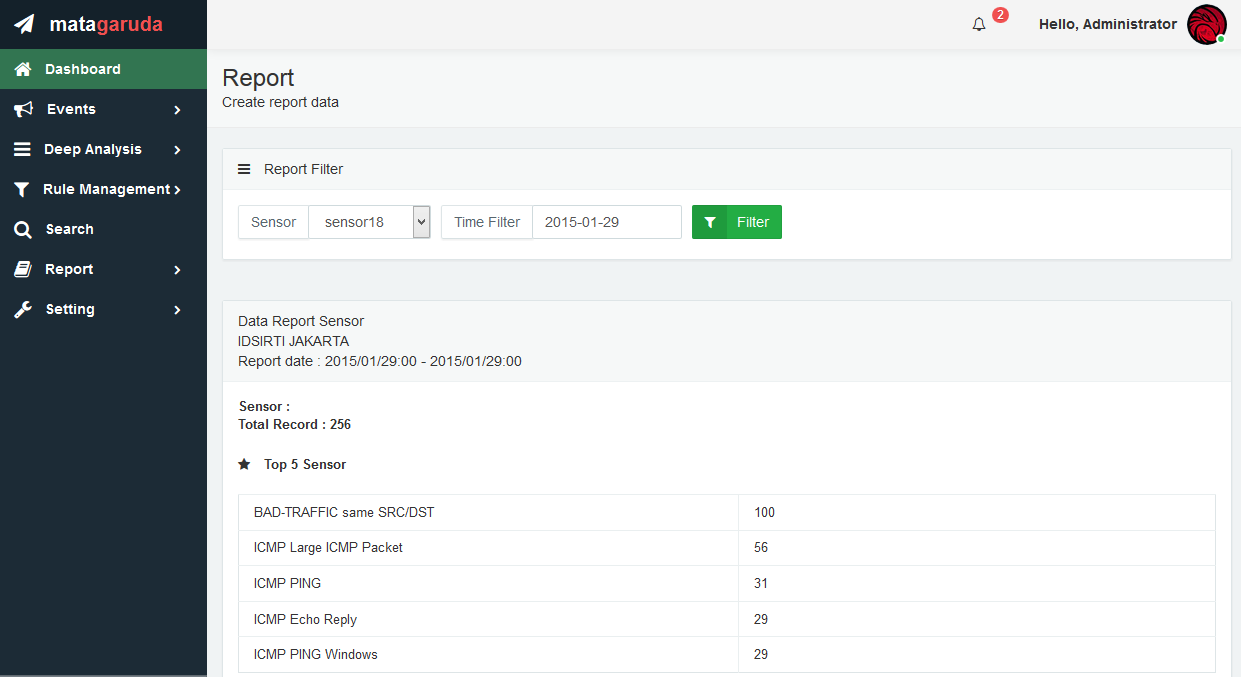


Gambar diatas menampilkan *pie chart* untuk masing-masing *protocol TCP/UDP* dan juga presentase jumlah perbandingan kedua *port* tersebut.

### **Pengujian Menu *Report***

Pada tahap ini menguji aplikasi apakah mampu menampilkan data transaksional untuk kebutuhan *report/*laporan atau tidak.





Dua gambar diatas menunjukkan hasil yang berbeda-beda untuk data *report* masing-masing *sensor* yang terhubung ke aplikasi Mata Garuda.

# **BAB V**

# **PENUTUP**

Berdasarkan analisa dari pengujian sistem IDS Snort dan Aplikasi Mata Garuda, penulis mendefenisikan beberapa kesimpulan dan saran yang mungkin berguna untuk pengembangan lebih lanjut.

## **Kesimpulan**

## **Saran**

**DAFTAR PUSTAKA**

1. Pilli, Emmaniel S., R.C. Joshi, Rajdeep Niyogi. *Network Forensic Frameworks: Survey and Research Challenges. India: Department of Electonics and Computer Engineering, Indian Institute Technology Roorke.* Retrieved Januari 2015, from http://www.sciencedirect.com/science/article/pii/S1742287610000113
2. Danyliw, Roman. *ACID*: *Analysis Console for Intrution Databases.* Retrieved Desember 2014, from http://academic.research.microsoft.com/Publication/2036125/acid-analysis-console-for-intrusion-databases.
3. Danyliw, Roman. *Analysis Console for Intrution Databases.* Retrieved Desember 2014, from http://academic.research.microsoft.com/Publication/2036125/acid-analysis-console-for-intrusion-databases.
4. [What and Why of Snorby · Snorby/snorby Wiki · GitHub](https://github.com/Snorby/snorby/wiki/What-and-Why-of-Snorby). Retrieved Desember 2014, from https://github.com/Snorby/snorby/wiki/What-and-Why-of-Snorby.

# **BIODATA PENULIS**

|  |  |  |
| --- | --- | --- |
|  | | |
|  | | |
| **Nama** | : | Gerry Italiano Wowiling |
| **NRP** | : | 2110135027 |
| **TTL** | : | Solo, 25 Mei 1990 |
| **Agama** | : | Kristen Protestan |
| **Alamat** | : | Sitoluama – Laguboti, Sumatera Utara |
| **No Hp** | : | +6285275347835 |
| **E-Mail** | : | gerry.wowiling@it.student.pens.ac.id |
|  |  | gerry\_vermouth@yahoo.co.id |
| **Hobi** | : | Olahraga, Membaca, Gaming |
|  |  |  |
| **Pendidikan** | : |  |
| 2013 – 2015 | : | D4 Lanjut Jenjang Teknik Informatika - Politeknik Elektronika Negeri Surabaya |
| 2008 – 2011 | : | D3 Teknik Komputer -  Politeknik Informatika Del Sumatera Utara |
| 2005 – 2008 | : | SMK Telkom Sandhy Putra Medan |

**LAMPIRAN**