|  |  |  |
| --- | --- | --- |
| Topik Capstone | **Topik Capstone** | |
| Siklus / Tahun | **Gasal atau Genap / (tahun)** | |
| Judul Dokumen | **Capstone TA**  Judul Capstone Proyek kelompok | |
| Jenis Dokumen | **IMPLEMENTASI PRODUK**  Catatan: Penggunaan dan penyebaran dokumen ini dikendalikan oleh Departemen Teknik Komputer Universitas Diponegoro | |
| Nomor Dokumen | **C400.[NoRev]TA[tahun].[1/2].[KodeKelompok]** | |
| Nomor Revisi | **NoRev** | |
| Nama File | **KodeKelompok.doc** | |
| Tanggal Penerbitan | **Tanggal Penerbitan** | |
| Unit Penerbit | **Departemen Teknik Komputer Universitas Diponegoro** | |
| Jumlah Halaman | **Jumlah Halaman** | Tidak termasuk sampul |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Pengusul** | | | | |
| Pengusul | Nama |  | Jabatan | Anggota |
|  | NIM |  |  |  |
|  | Tanggal |  | Tanda Tangan |  |
|  | Nama |  | Jabatan | Anggota |
|  | NIM |  |  |  |
|  | Tanggal |  | Tanda Tangan |  |
|  | Nama |  | Jabatan | Anggota |
|  | NIM |  |  |  |
|  | Tanggal |  | Tanda Tangan |  |
| Pembimbing 1 (Utama) | Nama |  | Tanda Tangan |  |
|  | Tanggal | **NIP.** |  |  |
| Pembimbing 2 | Nama |  | Tanda Tangan |  |
|  | Tanggal | **NIP.** |  |  |

|  |  |
| --- | --- |
| Versi, Tanggal, Oleh | Perbaikan |
|  |  |

Daftar Isi

[1. Pendahuluan 4](#_Toc188028257)

[1.1. Ringkasan isi dokumen 4](#_Toc188028258)

[1.2. Aplikasi Dokumen 4](#_Toc188028259)

[1.3. Referensi 4](#_Toc188028260)

[1.4. Daftar Singkatan 4](#_Toc188028261)

[2. Implementasi 4](#_Toc188028262)

[2.1. Implementasi Produk 4](#_Toc188028263)

[2.2. Tampilan Produk 13](#_Toc188028267)

[2.3. Demonstrasi Produk 16](#_Toc188028270)

[2.4. Kesimpulan Realisasi Desain Produk dengan Implementasi Produk 17](#_Toc188028271)

# **Pendahuluan**

## **Ringkasan isi dokumen**

*Isi Ringkasan Dokumen*

## **Aplikasi Dokumen**

*Dokumen ini berlaku berfungsi untuk menjelaskan:*

1. *Tahap implementasi dari rancangan pada Dokumen C300*
2. *Acuan tahap pengujian pada dokumen selanjutnya*
3. *Catatan proses pengerjaan dan revisi yang sedang berlangsung*

## **Referensi**

## **Daftar Singkatan**

# **Implementasi**

## **Implementasi Produk**

*Jelaskan dengan singkat gambaran implementasi produk yang dibuat.*

**Contoh:**

Implementasi LunaSystem Sistem *Monitoring* dan Deteksi Serangan Slow DoS/DRDoS Berbasiskan *Ensemble Learning* menggunakan ELK Stack pada Ditreskrimsus Polda Jateng Semarang dilakukan dengan pembuatan sistem yang dibagi berdasarkan 2 bagian, yaitu pembuatan model ensemble dengan menggunakan dataset CIC-IDS2017, CICCSEIDS2018, dan CICDDOS2019 setelah itu dilanjut dengan pembuatan *Log Management System* yang terintegrasi dengan *Web Application*.

Pembuatan model ensemble dibagi menjadi beberapa tahap yaitu, tahap *preprocessing data,* tahap *feature engineering,* dan yang terakhir adalah tahap pelatihan dan pengujian. Tahap *preprocessing* adalah langkah yang paling awal dilakukan, data mentah yang diambil dari dataset diubah dilakukan beberapa langkah, seperti menghapus data *redundant* yang tidak perlu dan menormalisasi data, sehingga data tersebut siap digunakan untuk dilatih dan diuji. Tahap berikutnya yaitu melakukan *feature engineering.* Pada tahap ini dilakukan dua tahap *feature engineering* yaitu melakukan *feature selection* dengan menggunakan *information gain* dan *Fast Correlation Base Filter*. Feature selection dengan menggunakan information gain adalah metode

A screenshot of a computer

Description automatically generateduntuk memilih fitur-fitur yang paling relevan dalam dataset berdasarkan seberapa baik fitur-fitur tersebut memisahkan data ke dalam kategori-kategori yang berbeda. *Fast Correlation-Based Filter (FCBF)* adalah metode *feature selection* yang efisien untuk memilih fitur-fitur paling relevan dari dataset. FCBF mengurangi redundansi dengan mempertahankan hanya fitur-fitur yang memiliki korelasi signifikan dengan target (label) yang ingin diprediksi. FCBF menghitung *C-correlation* (korelasi antara fitur dan kelas) dan *F-correlation* (korelasi antara dua fitur), kemudian menyaring fitur-fitur yang *C-correlation-*nya kurang dari ambang batas yang telah ditentukan dan menghapus fitur-fitur redundant dengan menggunakan Markov blanket yang telah diidentifikasi[[1]](https://www.zotero.org/google-docs/?Kj5GGG). Masuk ke tahap pelatihan dan pengujian. Pada tahap ini dataset dibagi menjadi 70 persen untuk data latih dan 30 persen untuk data uji. Hasil atau model dari tahap pelatihan kemudian disimpan dalam bentuk joblib. Model ini akan digunakan untuk memprediksi trafik secara *real time*.

Gambar 2.1. *Service* pcap2ciclog

A screenshot of a computer

Description automatically generatedGambar 2.1. menunjukkan sebuah *service* pcap2ciclog yang berguna untuk melakukan penangkapan trafik menggunakan TCPDump. *Service* tersebut akan berjalan selama t detik, kemudian menyimpan trafik tersebut dalam bentuk berkas .pcap. *Service* tersebut juga akan mengkonversi berkas pcap yang sudah dihasilkan ke bentuk berkas csv berdasarkan format *dataset* yang digunakan.

Gambar 2.2. *Service* Luna

Setelah berkas pcap berhasil dikonversi ke bentuk csv, berkas csv yang berisi trafik tersebut akan dilakukan data *preprocessing* dan *feature engineering* oleh *service Luna*, sama halnya seperti yang dilakukan pada tahap pembuatan model. Setelah itu *service Luna* akan melakukan prediksi terhadap trafik tersebut. Hasil prediksi akan menghasilkan berkas berbentuk csv dengan nama log.csv.

Konfigurasi Filebeat untuk memantau *log* dari *file* CSV yang berisi data Slow DoS dan mengirimkannya ke Logstash. Ini memungkinkan data yang relevan untuk dianalisis lebih lanjut atau diproses oleh sistem lain dalam infrastruktur.

Lakukan instalasi pada *server* yang akan dipantau, dan persiapkan *file* CSV yang berisi data DoS yang akan dimonitor. Selanjutnya melakukan konfigurasi pada Filebeat, lakukan konfigurasi pada *file* filebeat.yml. Di dalam filebeat.yml masukkan konfigurasi yang diperlukan seperti melakukan *setting paths* agar memungkinkan *user* untuk menentukan sumber data *log* yang akan dipantau dengan fleksibilitas, sehingga memastikan bahwa data *log* yang relevan dapat diakses dan diproses secara efisien oleh sistem pengolahan data. Konfigurasi dilakukan dengan tujuan untuk menentukan *input log* dari *file* CSV.

A screenshot of a computer

Description automatically generatedSetelah selesai melakukan konfigurasi pada Filebeat, simpan *file* konfigurasi dan lakukan *restart* pada layanan Filebeat. Periksa *log* Filebeat untuk memastikan tidak ada kesalahan dan data dari *file* CSV telah dikirimkan ke Elasticsearch dengan sukses.

Gambar 2.3. Konfigurasi Filebeat

Dalam menggunakan Filebeat, kita harus melakukan konfigurasi terlebih dahulu pada beberapa bagian. Agar *log file* dapat terbaca oleh Filebeat, lakukan konfigurasi pada bagian *path* dan sesuaikan dengan lokasi *log file*. Pada bagian *enabled* atur menjadi true agar *input* filebeat dapat terbaca dan bisa dieksekusi oleh sistem.

A screenshot of a computer

Description automatically generated

Gambar 2.4. Konfigurasi Koneksi ke Logstash

Pada gambar tersebut menjelaskan konfigurasi pada filebeat.yml. Konfigurasi *endpoint* Kibana dan *output* Logstash merupakan bagian penting dalam pengaturan sistem *monitoring* dan manajemen *log* menggunakan ELK stack (Elasticsearch, Logstash, dan Kibana). Pada konfigurasi Kibana, *host* Kibana diatur untuk berjalan di `localhost` pada *port default* `5601`, yang menunjukkan bahwa *dashboard* dan antarmuka pengguna Kibana akan diakses melalui http://localhost:5601. ID ruang Kibana tidak diatur, sehingga ruang *default* digunakan untuk mengelompokkan dan mengelola *dashboard* serta visualisasi.

Pada bagian *output* Logstash, *host* Logstash diatur untuk menerima *input* dari

`localhost` pada *port* `5044`, memungkinkan Beats seperti Filebeat untuk mengirim data *log* ke Logstash. Konfigurasi ini juga mencakup pengaturan opsional untuk mengaktifkan SSL demi keamanan, dengan opsi untuk menentukan sertifikat otoritas *root*, sertifikat SSL klien, dan kunci sertifikat klien untuk memastikan komunikasi yang aman antara Beats dan Logstash. Selain itu, bagian prosesor mengatur berbagai pemrosesan yang dapat diterapkan pada data sebelum dikirim ke *output*. Prosesor ini berguna untuk memodifikasi, menghapus, atau menambahkan data dalam *log* sesuai kebutuhan, meskipun dalam contoh ini tidak ada pemrosesan spesifik yang diatur. Keseluruhan konfigurasi ini mengarahkan data dari Beats ke Kibana dan Logstash, di mana Kibana digunakan untuk visualisasi data dan Logstash digunakan untuk memproses serta mengirim data ke Elasticsearch untuk penyimpanan lebih lanjut. Pengaturan SSL yang opsional memberikan opsi keamanan tambahan untuk komunikasi antar komponen.

### **Konfigurasi Logstash**

Konfigurasi Logstash bertujuan untuk mengonsumsi data yang dikirim dari Elasticsearch dan melakukan pemrosesan data yang sesuai dengan kebutuhan, seperti *parsing*, pengubahan format, atau penyimpanan ke penyimpanan data yang lain. Ini memungkinkan integrasi yang mulus antara Elasticsearch dan Logstash, memungkinkan aliran data yang lancar dan analisis yang efektif.

Logstash berfungsi sebagai titik tengah dalam arsitektur pengolahan data *log* yang melibatkan Filebeat dan Elasticsearch. Dengan menerima, memproses, dan mengirimkan data *log*, Logstash memainkan peran penting dalam mengelola aliran data dari sumber ke penyimpanan dan analisis. Fungsi utamanya meliputi:

1. **Menerima Data** : Logstash menerima data *log* yang dikirim oleh Filebeat.
2. **Pemrosesan** : Data *log* diproses menggunakan *filter* yang ditentukan, seperti pemetaan, *parsing*, pengubahan format, atau pengayaan data.
3. **Pengiriman** : Data *log* yang telah diproses dikirim ke Elasticsearch untuk penyimpanan dan analisis lebih lanjut.
4. **Integrasi** : Logstash menyediakan integrasi yang kuat dengan berbagai sumber data dan penyimpanan, memungkinkan pengolahan data yang fleksibel dan skalabel.

Dengan mengkonfigurasi Logstash untuk memproses data *log*, kita memastikan bahwa data yang dikumpulkan dari berbagai sumber dapat diproses, diperkaya, dan disimpan dengan efisien sebelum disajikan untuk analisis lebih lanjut. Ini membantu dalam pengelolaan aliran data *log* dalam lingkungan yang kompleks dan memfasilitasi pemahaman yang lebih baik tentang sistem dan aplikasi yang dipantau.

Konfigurasi Logstash ini dirancang untuk menerima, memproses, dan mengirim data ke Elasticsearch sambil juga mencetak data tersebut ke stdout untuk keperluan *debugging*. Pada bagian *input*, Logstash dikonfigurasi untuk menerima data dari *plugin* Beats pada *port* 5044. Beats adalah kumpulan pengirim data ringan seperti Filebeat dan Metricbeat yang mengirim data dari berbagai sumber ke Logstash.

A screenshot of a computer

Description automatically generatedPada bagian *filter*, Logstash memproses data yang diterima dalam format CSV. *Filter* CSV digunakan untuk memisahkan data berdasarkan koma dan memetakan nilai ke kolom yang telah didefinisikan seperti "Dst IP", "Src IP", "Src Port", "Dst Port", dan "Label". Setelah data diproses, bagian *output* mengirim data ke dua tujuan: stdout dan Elasticsearch. *Output* stdout digunakan untuk mencetak data ke konsol dengan format rubydebug yang mudah dibaca untuk tujuan *debugging*. Sementara itu, *output* Elasticsearch mengirim data yang telah diproses ke *instance* Elasticsearch yang berjalan pada *localhost* pada *port* 9200 dan menyimpannya di indeks bernama "sekarangta". Konfigurasi ini memungkinkan data yang diterima dapat dengan mudah dianalisis dan divisualisasikan menggunakan Elasticsearch dan Kibana, memberikan wawasan yang lebih dalam tentang data tersebut.

Gambar 2.5. Konfigurasi Logstash

### **Konfigurasi Elastisearch**

Konfigurasi Elasticsearch bertujuan untuk menyimpan data log yang diproses oleh Logstash dan memfasilitasi penganalisisan data log menggunakan Kibana. Ini membentuk bagian penting dari ELK Stack (Elasticsearch, Logstash, Kibana) yang digunakan untuk pemantauan dan analisis *log*.

Elasticsearch berperan sebagai mesin penyimpanan dan pencarian yang kuat dalam arsitektur ELK Stack. Fungsinya meliputi:

1. **Penyimpanan Data**: Elasticsearch menyimpan data *log* yang diproses oleh Logstash dalam bentuk indeks yang terstruktur dan terdistribusi.
2. **Pencarian dan Penganalisisan**: Elasticsearch menyediakan API pencarian dan penganalisisan yang kuat untuk melakukan kueri dan analisis data *log* dengan cepat dan efisien.
3. **Skalabilitas**: Elasticsearch dirancang untuk skalabilitas horizontal, memungkinkan penambahan *node* dan partisi untuk menangani volume data yang besar.
4. **Integrasi dengan Kibana**: Elasticsearch berintegrasi langsung dengan Kibana, memungkinkan visualisasi dan analisis data *log* yang mudah dan intuitif.

Dengan mengkonfigurasi Elasticsearch untuk menyimpan data *log* dan memfasilitasi pencarian dan analisis data, kita menciptakan fondasi yang kuat untuk pemantauan dan analisis *log* yang efektif. Elasticsearch memainkan peran sentral dalam infrastruktur ELK Stack, memungkinkan pengolahan dan pemahaman yang lebih baik tentang data *log* dari berbagai sumber. Dengan analisis yang kuat dan cepat, kita dapat dengan cepat mengidentifikasi masalah, mendiagnosa penyebabnya, dan mengambil tindakan yang sesuai untuk meningkatkan kinerja dan keamanan sistem.

Konfigurasi Elasticsearch dalam *file* elasticsearch.yml mengatur beberapa aspek penting dari *server* Elasticsearch. Pengaturan network.host: 0.0.0.0 menginstruksikan Elasticsearch untuk mendengarkan koneksi pada semua antarmuka jaringan yang tersedia, memungkinkan akses dari semua alamat IP yang terikat pada mesin tersebut. Ini memberikan fleksibilitas untuk menerima koneksi dari berbagai jaringan atau perangkat. Pengaturan http.port: 9200 menetapkan *port* 9200 sebagai *port* di mana Elasticsearch akan mendengarkan koneksi HTTP, yang merupakan *port default* untuk layanan Elasticsearch dan digunakan untuk operasi pencarian, pengindeksan, dan administrasi data. Selanjutnya, discovery.seed\_hosts: [] menunjukkan bahwa tidak ada *host seed* yang ditentukan untuk penemuan kluster, yang mungkin berarti *instance* Elasticsearch ini beroperasi sebagai *node* tunggal atau menggunakan metode lain untuk penemuan *node* dalam kluster. Terakhir, http.host: 0.0.0.0 memastikan bahwa Elasticsearch mendengarkan koneksi HTTP pada semua antarmuka jaringan, memberikan aksesibilitas yang luas untuk layanan HTTP-nya.

A screenshot of a computer

Description automatically generatedMeskipun pengaturan ini memberikan fleksibilitas tinggi dan kemudahan akses, terutama dalam lingkungan pengembangan atau pengujian, penting untuk mempertimbangkan langkah-langkah keamanan tambahan, seperti konfigurasi Firewall dan penggunaan autentikasi serta enkripsi TLS/SSL, terutama dalam lingkungan produksi, untuk melindungi *instance* Elasticsearch dari akses yang tidak sah.

Gambar 2.6. Konfigurasi Elasticsearch

### **Konfigurasi dan Visualisasi menggunakan Kibana**

Proses visualisasi *log* serangan Slow DoS di Kibana dalam ELK Stack dimulai dengan menyiapkan data *log* yang telah dikirim dan tersimpan dalam Elasticsearch, dilanjutkan dengan pembuatan *index pattern* di Kibana untuk mengakses data *log* tersebut, kemudian pengguna dapat membuat berbagai jenis visualisasi seperti histogram, grafik batang, atau peta untuk menampilkan informasi penting seperti pola serangan, jumlah serangan per IP sumber, atau distribusi geografis, yang kemudian disusun dalam *dashboard* untuk memungkinkan pemantauan serangan Slow DoS secara *real*-*time*, sehingga dengan visualisasi ini, pengguna dapat mengidentifikasi tren serangan, memahami dampaknya terhadap sistem, serta mengambil tindakan yang tepat untuk meningkatkan keamanan dan kinerja sistem secara keseluruhan dengan memberikan pemahaman yang lebih baik tentang pola dan karakteristik serangan Slow DoS.

A screenshot of a computer

Description automatically generated

Gambar 2.7. Visualisasi *Discover* Kibana

## **Tampilan Produk**

Tampilkan screenshot dari produk yang dibuat untuk masing-masing role/fungsi yang didesain untuk produk tersebut. Tampilan bisa dibuat per fungsi atau per kelompok user.

Contoh:

#### **Web Application**

### **Halaman Login**

A screenshot of a computer

Description automatically generated

Gambar 2.8. Tampilan Login

Gambar 2.8 merupakan tampilan dari masuk atau *login* web *application* Luna System, halaman ini berisikan *text field* dan button *login* untuk masuk pada halaman *admin*. Admini diminta memasukkan *username* dan *password* yang telah tersedia.

### **Demonstrasi Produk**

A screenshot of a computer

Description automatically generated

Gambar 2.9. Tampilan *Dashboard* LunaSystem

Pada gambar 2.9 merupakan tampilan dari *dashboard* setelah sebelumnya *user* melakukan *login* ke dalam sistem. Di dalam menu *Dashboard* terdapat beberapa *chart* dan *table* informasi dari serangan siber yang terdeteksi. Pada *line chart* terdapat hasil deteksi serangan dalam kurun waktu tiap 10 detik, ini akan membuat perubahan secara *realtime*. Selanjutnya terdapat *table summary* dari hasil serangan siber yang terdeteksi, tabel ini akan berisi berisi *source* IP, *destination* IP, dan label jenis serangan. Kemudian pada *chart Source* IPberfungsi untuk menampilkan IP dari *server* yang mengalami serangan siber.

A screenshot of a graph

Description automatically generated

Gambar 2.10. Tampilan Menu *Distribution Attack*

A screenshot of a graph

Description automatically generatedPada gambar 2.10 tersebut, merupakan tampilan dari menu *distribution attack*, dalam menu ini akan terdapat 3 *chart* dan 1 *table*. Pada *chart top* IP *attacker* akan menampilkan total dan IP dari IP si penyerang. Selanjutnya terdapat *chart top* IP *targets*, *charts* ini berfungsi untuk menampilkan total dari IP yang menerima serangan dari IP *attacker*. Kemudian terdapat *donut charts*, *donut charts* ini untuk menampilkan total data dari jenis serangan yang telah terdeteksi. Terakhir terdapat *alerts summary*, dimana tabel ini akan menampilkan hasil serangan siber yang terdeteksi, tabel ini akan berisi berisi *source* IP, *destination* IP, dan label jenis serangan.

Gambar 2.11. Tampilan menu IP *Distribution*

A screenshot of a graph

Description automatically generatedGambar 2.11 diatas merupakan tampilan menu dari IP *Distribution*, kemudian terdapat *chart* untuk menampilkan IP sumber dari mana serangan berasal. Selain itu juga terdapat *chart* IP *destination*, ini akan menunjukkan IP tujuan yang paling sering diserang. Ini dapat mengidentifikasi *server* atau perangkat mana yang menjadi target utama.

Gambar 2.12. Tampilan menu Port Distribution

Pada gambar 2.12 tersebut, merupakan tampilan dari menu *port distribution*, pada menu ini terdapat 2 chart dan 2 table. *chart* ini untuk menunjukkan *port* sumber yang paling sering digunakan dalam serangan. Selain itu juga terdapat chart *port destination*, ini akan menunjukkan *port* tujuan yang paling sering diserang. Ini dapat mengidentifikasi *server* atau perangkat mana yang menjadi target utama. Selanutnya terdapat *table* untuk menampilkan kesimpulan dari *port* penyerang dan *port* yang diserang, tabel ini akan mencakup *port*, label serangan dan total serangan yang terjadi.

## **Demonstrasi Produk**

*Sediakan link dan QR code yang merujuk ke video demonstrasi produk.*

**Contoh:**

Link : https://drive.google.com/drive/folders/13rugbQGNDQa\_HG53q4kCHoVBwJvy04YO

?usp=sharing

## **Kesimpulan Realisasi Desain Produk dengan Implementasi Produk**

*Di dalam sub bab ini, Tim Capstone harus membuat tabel kesimpulan untuk melakukan pelacakan dari list desain yang ada apakah ada proses implementasi yang gagal dari dokumen desain yang ada serta melakukan menjabarkan analisis dari kegagalan yang ada.*