TUGAS AKHIR – KI141502

EVALUASI SISTEM PENDETEKSI INTRUSI BERBASIS ANOMALI DENGAN N-GRAM DAN INCREMENTAL LEARNING

I MADE AGUS ADI WIRAWAN
NRP 5112 100 036

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

Dosen Pembimbing II
Baskoro Adi Pratomo, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESES – KI141502

EVALUATION OF ANOMALY BASED INTRUSION DETECTION SYSTEM WITH N-GRAM AND INCREMENTAL LEARNING

I MADE AGUS ADI WIRAWAN
NRP 5112 100 036

Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

Supervisor II
Baskoro Adi Pratomo, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

LEMBAR PENGESAHAN

EVALUASI SISTEM PENDETEKSI INTRUSI BERBASIS ANOMALI DENGAN N-GRAM DAN INCREMENTAL LEARNING

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
I MADE AGUS ADI WIRAWAN
NRP : 5112 100 036

Disetujui oleh Dosen Pembimbing Tugas Akhir :

ROYYANA MUSLIM IJTIHADIE, S.Kom., M.Kom., Ph.D. ..... (pembimbing 1)
NIP: 19770824 202624 1 001

BASKORO ADI PRATOMO, S.Kom., M.Kom. ..... (pembimbing 2)
NIP: 19870218 201404 1 001

SURABAYA
JULI, 2016

EVALUASI SISTEM PENDETEKSI INTRUSI BERBASIS ANOMALI DENGAN N-GRAM DAN INCREMENTAL LEARNING

Nama Mahasiswa	**: I MADE AGUS ADI WIRAWAN**
NRP	**: 5112100036**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**
Dosen Pembimbing 2	**: Baskoro Adi Pratomo, S.Kom., M.Kom.**

Abstrak

Keberadaan teknologi informasi yang terus berkembang dengan pesat menjadikan kebutuhan akan penggunaannya semakin hari semakin meningkat. Transaksi data melalui internet telah menjadi kebutuhan wajib hampir dari semua perangkat lunak yang ada saat ini. Perangkat lunak seperti media social, colud server, online game, aplikasi layanan pemerintah, aplikasi pengontrol suatu tempat secara remote, dsb. Tentu dengan berbagai macam penggunaan internet tersebut dibutuhkan metode untuk mengamankan jaringannya.

Sistem pendeteksi intrusi atau yang pada umumnya disebut IDS (Intrusion Detection System) merupakan solusi untuk mengamankan suatu jaringan. Sistem ini nantinya bertugas untuk menentukan apakah suatu paket merupakan bentuk serangan atau paket biasa sesuai dengan kondisi tertentu. Saat ini telah banyak dikembangkan aplikasi IDS (Intrusion Detection System), namun sebagian besar yang dikembangkan berbasis signature atau menggunakan rule, dan sebagaian kecil menggunakan anomali. Anomali adalah suatu metode untuk mencari penyimpangan dalam sebuah data.

Pada aplikasi ini konsep IDS yang diterapkan adalah IDS berbasis anomali dimana analisis datanya pada infromasi paket data yang dikirimkan. Pada tugas akhir ini menggunakan dua metode, yaitu metode n-gram yang digunakan untuk mengitung distribusi byte karakter pada paket data sedangkan metode

mahalanonis distance digunakan untuk menghitung jarak antara paket data normal dan paket data yang berupa intrusi.

Metode mahalanobis distance dapat membedakan paket data yang normal dan paket data yang berupa intrusi dengan menghitung rata-rata dan standar deviasi dari paket data.

Kata kunci : N-Gram, Mahalanobis Distance, Incremental Learning

EVALUATION OF ANOMALY BASED INTRUSION DETECTION SYSTEM WITH N-GRAM AND INCREMENTAL LEARNING

Student's Name	**: I MADE AGUS ADI WIRAWAN**
Student's ID	**: 5112100036**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**
Second Advisor	**: Baskoro Adi Pratomo, S.Kom., M.Kom.**

Abstract

The rapid development of information technology is inevitable wich made its necessity is growing every single day. Data transaction through internet has become the primary need of most software nowadays. Software like social media, cloud server, online game, e-government, remote application, etc. With the various needs of the internet, it is obvious that we need a method that can guarantee its safety.

IDS which stands for Intrusion Detection System is the solution to protect the internet network. This system will decide wether a packet is safe or dangerous for the network depends on certain condition. Nowadays many IDS (Intrusion Detection System) has been developed, but most are developed base signature or use the rule, and a small part sing anomaly. Anomaly is a method to look for irregularities in the data.

In this application IDS concept that is applied is based anomaly in which the data analysis on the data packets transmitted. In this thesis using two methods, the n-gram method used to calculate the distribution of byte character data paket while the mahalanobis distance methods used to calculated the distance between the normal data packets and intrusion data packets.

Mahalanobis distance methods can distinguish between normal data packets and intrusion data packets by calculating the average and standar deviation of the data packets.

Keyword : N-Gram, Mahalanobis Distance, Incremental Learning

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam Tugas Akhir ini yang meliputi latar belakang, perumusan masalah, batasan, tujuan dan manfaat pembuatan Tugas Akhir serta metodologi dan sistematika pembuatan buku Tugas Akhir ini. Dari uraian dibawah ini diharapkan gambaran Tugas Akhir secara umum dapat dipahami dengan baik.

1.1 Latar Belakang

Semakin pesatnya perkembangan teknologi informasi memudahkan orang-orang untuk saling tukar menukar data baik melalui internet maupun intranet. Tentunya dengan mudahnya berbagi data itulah sangat memungkinkan terjadinya serangan terhadap data tersebut terutama melalui jaringan komputer. Sistem pendeteksi intrusi atau yang pada umumnya disebut IDS (*Intrusion Detection System*) merupakan senjata utama untuk mengamankan suatu jaringan dimana sistem ini nantinya bertugas untuk mengidentifikasi dan mencatat apakah suatu paket data tersebut merupakan bentuk serangan atau paket data bisa.

Saat ini telah banyak dikembangkan aplikasi IDS (*Intrusion Detection System*), namun sebagian besar yang dikembangkan berbasis *signature* atau menggunakan *rule*, dan sebagian kecil menggunakan *anomaly*. *Anomaly* pada dasarnya adalah mencari data yang menyimpang dari sekumpulan data normal. IDS yang berbasis pada *anomaly* bersifat lebih fleksibel, karena dapat mengenali pola serangan baru tanpa harus memperbaharui basis data pola serangan. IDS yang berbasis pada anomali memiliki sebuah kecerdasan buatan yang mampu mendeteksi dan mengenali sebuah serangan. IDS yang berbasis anomali menggabungkan metode analisis dan statistik untuk mengenali penyimpangan tersebut. Kelemahan dari metode ini adalah kemungkinan salah identifikasi pada data yang diolah.

Sistem kerja intrusi ini pada dasarnya dikirimkan lewat jaringan dengan paket-paket data yang sama dengan paket data normal. Dengan banyaknya paket data yang masuk kedalam sebuah host, tentunya host ini harus bisa mengenali paket data, apakah paket data tersbut terdapat paket data yang berupa intrusi atau tidak. Hal tersebut dapat dikenali dengan cara mengelompokkan data berdasarkan beberapa hal yang membedakan antara paket data normal dengan paket data yang berupa intrusi.

Maka untuk membedakan hal tersebut diperlukan sebuah sistem deteksi intrusi dimana nantinya sistem deteksi intrusi tersebut menggunakan gabungan metode analisis dan statistik yang berfungsi mengenali perbedaan paket data normal maupun paket data berupa intrusi. Selain itu, sistem deteksi intrusi yang dapat mempelajari paket data normal yang baru sebagai data *training*.

Untuk dapat menghitung jarak mahalanobis dari paket data, diperlukan metode yang dapat merubah informasi paket data menjadi nilai yang dapat dihitung. Metode n-gram dapat digunakan untuk mebuat model yang sederhana dan cepat untuk dihitung khususnya menghitung distribusi karakter pada suatu paket data. N-Gram merupakan metode yang paling efesien dan efektif dalam membuat model dari suatu paket data.

Metode mahalanobis *distance* berguna untuk membedakan paket-paket data berdasarkan anomali yang terjadi. Untuk dapat mempelajari paket data normal yang baru menggunakan metode *incremental learning*, dimana metode ini nantinya memperbaharui rata-rata dan standar deviasi dari model paket data yang ada pada data *training*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana membangun sistem deteksi intrusi yang dapat membaca data set dari DARPA IDS tahun 1999 data set?

2. Bagaimana membangun sistem deteksi intrusi yang dapat menangkap paket data dari *network interface* suatu komputer?
3. Bagaimana menerapkan metode n-gram pada konten paket data?
4. Bagaimana cara mengklasifikasikan paket data menjadi dua kelompok, yaitu paket data normal dan paket data yang berupa intrusi dengan menggunakan metode Mahalanobis distance?
5. Bagaimana membangun sistem deteksi intrusi yang menerapkan metode *incremental learning*?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data set yang digunakan adalah DARPA IDS tahun 1999.
2. Jenis protokol yang akan diperiksa adalah TCP dengan port aplikasi dari FTP(21), Telnet(23), SMTP(25), HTTP(80) dan UDP dengan port aplikasi dari DNS Server(53).

1.4 Tujuan

Tujuan dari dibuatnya Tugas Akhir ini adalah membuat sistem pendeteksi intrusi yang mampu mengenali serangan pada lalu lintas jaringan dengan menggunakan metode Mahalanobis Distance berbasis anomali yang nantinya mampu membedakan paket data normal maupun paket data yang berupa intrusi.

1.5 Manfaat

Dengan dibuatnya Tugas Akhir ini akan memberikan konsep baru pada cara membedakan paket data normal dan paket data yang berupa intrusi.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini, yaitu membuat aplikasi deteksi intrusi pada jaringan komputer berbasi anomali dengan *n-gram* dan *incremental learning*.

2. Studi literatur
   Pada tahap ini dilakukan pemahaman informasi dan literatur yang diperlukan untuk pembuatan implementasi program. Tahap ini diperlukan untuk membantu memahami pengunaan komponen-komponen terkait dengan sistem yang akan dibangun, antara lain : IDS secara umum, IDS berbasis anomali, metode Mahalanobis *Distance*, metode *n-gram*, metode *incremental learning* dan Jpcap.

3. Analisis dan desain perangkat lunak
   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain fungsi yang akan dibuat yang ditunjukkan melalui diagram alir. Fungsi utama yang akan dibuat pada tugas akhir ini meliputi fungsi *sniffer*, rekonstruksi paket data, menghitung *n-gram* paket data, menghitung jarak mahalanobis dan *incremental learning*.

4. Implementasi perangkat lunak
   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga

menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi
   Pada tahapan ini dilakukan uji coba pada data yang telah dikumpulkan. Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian data dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan pada program.

6. Penyusunan buku Tugas Akhir.
   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir secara keseluruhan. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan
Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka
Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini. Dasar teori yang digunakan adalah IDS secara umum, IDS berbasis anomali, metode Mahalanobis *Distance*, metode *n-gram*, metode *incremental learning* dan Jpcap.

Bab III Desain dan Perancangan

Bab ini berisi tentang rancangan sistem yang akan dibangun dan disajikan dalam bentuk diagram alir. Fungsi utama yang akan dibuat pada tugas akhir ini meliputi fungsi *sniffer,* rekonstruksi paket data, menghitung *n-gram* paket data, menghitung jarak mahalanobis dan *incremental learning*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode program yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori yang menjadi dasar dari pembuatan Tugas Akhir ini. Teori yang dibahas mencakup elemen-elemen yang terkait dalam topik Tugas Akhir mulai dari sumber dari permasalahan, pendekatan yang digunakan, serta metode dan teknologi yang digunakan untuk pengerjaan Tugas Akhir ini.

2.1 IDS

IDS (*Intrusion Detection System*) [1] adalah aplikasi perangkat lunak yang digunakan untuk menyiapakan tindakan untuk menghadapi sebuah serangan. Hal ini dijalankan dengan cara mengumpulkan informasi dari berbagai sumber, baik dari suatu sistem maupun jaringan, lalu menganalisanya untuk menentukan ada tidaknya ancaman. IDS dikategorikan menjadi 3, yaitu [1]:

a. NIDS (*Network Intrusion Detection System*), menjalankan analisa terhadap *traffic* dalam sebuah *subnet*. Bekerja secara acak dan mencocokkannya dengan kumpulan *rule* serangan yang sudah disimpan pada *library*. Ketika NIDS berhasil mendeteksi serangan atau perilaku yang abnormal, peringatan akan dikirim ke administrator jaringan.

b. NNIDS (*Network Node Intrusion Detection System*), sedikit meirp dengan NIDS namun NNIDS hanya bekerja pada satu host saja tidak untuk satu jaringan. Contoh penggunaan NNIDS adalah pada VPN, yaitu dengan memeriksa *traffic* ketika sudah terdekripsi. Dengan cara ini dapat diketahui apakah seseorang sedang mencoba merusak sebuah VPN *server*.

c. HIDS (*Host Based Intrusion Detection System*), mengambil *snapshot* dari sistem yang dimiliki dan

mencocokkannya dengan *snapshot* yang sudah diambil sebelumnya. Bila sebuah *system files* penting telah termodifikasi atau terhapus, sebuah peringatan akan dikirimkan ke administrator. Contoh penggunaannya adalah pada sebuah mesin yang bersifat *mission critical*, sehingga tidak boleh ada perubahan terhadap konfigurasinya.

2.2 IDS Berbasis Anomali

Anomaly pada dasarnya adalah mencari sebuah data yang menyimpang dari sekumpulan data normal. IDS yang berbasis *anomaly* menggabungkan metode analisis dan statistik untuk mengenali penyimpangan tersebut [2]. IDS yang berbasis pada *anomaly* bersifat lebih fleksibel, karena dapat mengenali pola serangan baru tanpa harus memperbaharui basis data pola serangan. IDS yang berbais pada *anomaly* memiliki sebuah kecerdasan buatan yang mampu mendeteksi dan mengenali sebuah serangan.

Kelemahan dari metode anomali ini adalah kemungkinan terjadinya salah identifikasi pada data yang diolah, juga ada kemungkinan terjadi kesalahan pada data normal yang menyebabkan aplikasi tidak dapat mengenali serangan.

2.3 Jpcap

Jpcap [3] adalah kumpulan kelas-kelas Java yang menyediakan *interface* dan sistem untuk *packet capture* pada jaringan. Dengan bantuan Jpcap, para pengembang pada khususnya yang menggunakan bahasa pemrograman Java dapat membuat aplikasi yang memiliki kegunaan *packet capture*. Pada Gambar 2.1 adalah contoh penggunaan Jpcap pada suatu java *class*.

Gambar 2.1 Contoh penggunaan Jpcap

Salah satu kegunaan Jpcap yang sering dijumpai adalah *offline capture*. Pengertian *offline capture* adalah pembacaan *dump file* yang didapatkan dari hasil aktual suatu *data traffic*. *Offline capture* dilakukan untuk menganalisa *data traffic* yang sudah disimpan untuk tujuan tertentu antara lain, menganalisa kebiasaan pengguna, mendeteksi anomali yang mungkin terlewat dari pengawasan, dan lain sebagainya.

```
1  JpcapCaptor captor =
   JpcapCaptor.openFile("inside.tcpdump");
2  while(true){
3   Packet packet=captor.getPacket();
4   if(packet == null || packet == Packet.EOF) break;
5   System.out.println(packet);
6  }
7  captor.close();
```

Gambar 2.2 Kode sumber penggunaan Jpcap untuk *offline capture*

```
1399207224:213524
1399207224:266954 /10.151.36.24->/239.255.255.250 protocol(17) priority(0)  hop(1)
1900
1399207224:325480 /fe80:0:0:0:e10a:9351:be2f:5683->/ff02:0:0:0:0:0:1:2 protocol(17)
hop(1) UDP 546 > 547
1399207224:482300 /10.151.36.22->/10.151.36.3 protocol(17) priority(0)  hop(128)  c
1399207224:922008 /10.151.36.1->/224.0.0.10 protocol(88) priority(6)  hop(2)  offse
1399207225:181347 /10.151.36.29->/255.255.255.255 protocol(17) priority(0)  hop(128
> 17500
1399207225:183822 /10.151.36.29->/10.151.36.255 protocol(17) priority(0)  hop(128)
17500
1399207225:187066 /fe80:0:0:0:b0b8:8c4c:3dc0:d9d3->/ff02:0:0:0:0:0:1:2 protocol(17)
hop(1) UDP 546 > 547
```

Gambar 2.3 Contoh keluaran *offline capture*

2.4 N-Gram

Pada dasarnya, model N-Gram [4] adalah model probabilistik yang awalnya dirancang oleh ahli matematika dari Rusia pada awal abad ke-20 dan kemudian dikembangkan untuk memprediksi *item* berikutnya dalam urutan *item*. Item bisa berupa huruf / karakter, kata, atau yang lain sesuai dengan aplikasi. Salah satunya, model *n-gram* yang berbasis kata digunakan untuk memprediksi kata berikutnya dalam urutan kata tertentu. Dalam arti bahwa sebuah *n-gram* hanyalah sebuah wadah kumpulan kata dengan masing-masing memiliki panjang *n* kata. Sebagai contoh, sebuah *n-gram* ukuran 1 disebut sebagai *unigram;* ukuran 2 sebagai *bigram*; ukuran 3 sebagai *trigram*, dan seterusnya.

Pada pembangkitan karakter, *N-gram* terdiri dari *substring* sepanjang *n* karakter dari sejumlah *string* dalam definisi lain *n-gram* adalah potongan sejumlah *n* karakter dari sebuah *string*. Metode *n-gram* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah *n* dari sebuah kata secara kontinuitas dibaca dari teks sumber sehingga akhir dari dokumen. Sebagai contoh : kata "TEXT" dapat diuraikan ke dalam beberapa *n-gram* berikut :

uni-gram : T, E, X, T
bi-gram : TE, EX, XT
tri-gram : TEX, EXT
quad-gram : TEXT, EXT_
dan seterusnya.

Sedangkan pada pembangkit kata, metode *n-gram* ini digunakan untuk mengambil potongan kata sejmlah *n* dari sebuah rangkaian kata (kalimat, paragraf, bacaan) yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen. Sebagai contoh : kalimat "saya dapat melihat cahaya itu." Dapat diuraikan ke dalam beberapa *n-gram* berikut :

uni-gram : saya, dapat , melihat, cahaya, itu
bi-gram : saya dapat, dapat melihat, melihat cahaya, cahaya itu
tri-gram : saya dapat melihat, dapat melihat cahaya, melihat cahaya itu_

dan seterusnya.

Salah satu keunggulan menggunakan *n-gram* dan bukan suatu kata utuh secara keseluruhan adlah bahwa *n-gram* tidak terlalu sensitif terhadap kesalahan penulisan yang terdapat pada suatu dokumen.

2.5 Simplified Mahalanobis Distance

Mahalanobis distance [5] adalah sebuah metode statistika untuk menghitung jarak antara titik P dan distribusi D. Prinsip Mahalanobis *Distance* adalah mengitung jarak di ruang multidimensional antara sebuah pengamatan dengan pusat dari semua pengamatan. Pada Tugas Akhir ini Mahalanobis *Distance* digunakan untuk menghitung jarak antara distribusi *byte* karakter dari payload baru terhadap model yang ada pada data *training*. Semakin jauh jaraknya, semakin besar kemungkinan payload ini tidak normal.

Mahalanobis *distance* dari sebuah payload baru dapat dihitung jika sistem sudah mempunyai data *training*. Selanjutnya menghitung rata-rata dan standar deviasi dari model yang ada pada data *training*. Untuk menghitung rata-rata dari model yang ada pada data *training* dapat dilihat pada persamaan (2.2). Sedangkan untuk menghitung standar deviasi dari model yang ada pada data *training* dapat dilihat pada persamaan (2.4). Setelah selesai

menghitung rata-rata dan standar deviasi dari model yang ada pada data *training* baru dapat menghitung jarak mahalanobis dari payload baru dengan menggunakan persamaan (2.1). Format data kasar yang ada pada Mahalanobis *disatance* dapat dilihat pada Tabel 2.1.

$$d(x,\overline{y}) = \sum_{i=0}^{n-1} (|x_i - \overline{y_i}|/(\overline{\sigma_i} + \alpha)) \qquad (2.1)$$

dimana,

d = jarak mahalanobis
x_i = variable ke-i dari *payload* baru
$\overline{y_i}$ = rata-rata variable ke-i dari model data *training*
$\overline{\sigma_i}$ = standar deviasi variable ke-i dari model data *training*
α = *smoothing factor*

Tabel 2.1 Format data kasar didalam Mahalanobis *Distance*

	Variabel (karakteristik)						
Object	X_1	X_2	…	X_i	…	X_{p-1}	X_p
1	.	.	…	.	…	.	.
2	.	.	…	.	…	.	.
3	.	.	…	.	…	.	.
.	.	.	…	.	…	.	.
.	.	.	…	.	…	.	.
.	.	.	…	.	…	.	.
K	X_{k1}	X_{k2}	…	X_{ki}	…	$X_{k,p-1}$	$X_{k,p}$
.	.	.	…	.	…	.	.
.	.	.	…	.	…	.	.
.	.	.	…	.	…	.	.
N	X_{N1}	X_{N2}	…	X_{Ni}	…	$X_{N,p-1}$	$X_{N,p}$
Average	$\overline{X_1}$	$\overline{X_2}$	…	$\overline{X_i}$	…	$\overline{X_{p-1}}$	$\overline{X_p}$
Standar deviation	S_1	X_1	…	X_1	…	X_1	X_1

Persamaan untuk mencari rata-rata, yaitu:

$$\overline{X_i} = \frac{1}{N} \sum_{k=1}^{N} X_{ki} \quad (2.2)$$

dimana,

$\overline{X_i}$ = rata-rata variabel ke-i
N = jumlah object model
X_{ki} = nilai variabel ke-i

Persamaan untuk mencari nilai standar deviasi, yaitu:

$$S_i = \sqrt{\frac{\sum_{k=1}^{N} (X_{ki} - \overline{X}_i)^2}{N-1}} \quad (2.3)$$

dimana,

S_i = standar deviasi variabel ke-i
X_{ki} = nilai dari variabel ke-i
$\overline{X_i}$ = rata-rata variabel ke-i
N = jumlah object model

2.6 *Incremental Learning*

Incremental Learning merupakan proses untuk memperbaharui nilai rata-rata dan standar deviasi dari model yang ada pada data *training* ketika menambahkan payload baru. Proses ini diperlukan untuk meningkatkan akurasi dari setiap model ketika ditambah data sampel baru.

Untuk menghitung Mahalanobis *distance* versi *Incremental Learning* diperlukan rata-rata dan standar deviasi dari masing-masing karakter ASCII untuk setiap sampel baru yang dihitung. Untuk menghitung rata-rata dari sebuah karakter dapat dilihat pada persamaan (2.3). Selanjutnya agar dapat memperbaharui nilai rata-rata dari model yang ada pada data *training*, diperlukan jumlah sampel yang telah dihitung sebelumnya [6]. Untuk menghitung nilai rata-rata yang baru dapat dilihat pada persamaan (2.4).

Sedangkan untuk mengitung standar deviasi yang baru diperlukan rata-rata dari x_i^2 pada model sebelumnya. Untuk menghitung standar deviasi yang baru dapat dilihat pada persamaan (2.5).

Persamaan untuk menghitung rata-rata baru dari model yang diamati, yaitu:

$$\overline{\boldsymbol{x}} = \frac{\overline{\boldsymbol{x}} \times \boldsymbol{N} + \boldsymbol{x}_{\boldsymbol{N+1}}}{\boldsymbol{N+1}} = \overline{\boldsymbol{x}} + \frac{\boldsymbol{x}_{\boldsymbol{N+1}} - \overline{\boldsymbol{x}}}{\boldsymbol{N+1}} \qquad (2.4)$$

dimana,

$\overline{\boldsymbol{x}}$ = rata-rata baru
$\boldsymbol{x}_{\boldsymbol{N+1}}$ = nilai dari variabel yang baru
$\boldsymbol{N}$ = jumlah sampel sebelumnya

Persamaan untuk menghitung standar deviasi baru dari model yang diamati, yaitu:

$$\boldsymbol{S}_i = \sqrt{\frac{(\boldsymbol{n+1}) \times (\sum_{i=1}^{n} \boldsymbol{x}_i{}^2 + \boldsymbol{x}_{n+1}{}^2) - (\sum_{i=1}^{n} \boldsymbol{x}_i + \boldsymbol{x}_{n+1})^2}{(\boldsymbol{n+1})\boldsymbol{n}}} \qquad (2.5)$$

dimana,

S_i = standar deviasi variabel ke-i
x_i = nilai dari variabel ke-i
x_{n+1} = nilai dari variabel yang baru
n = jumlah object model

2.7 DARPA 1999

Subbab ini akan menjelaskan data set yang nantinya akan digunakan untuk data *training* dan menguji aplikasi ini. Data set yang digunakan adalah DARPA 1999 [7]. Data set ini berisi paket-paket hasil tangkapan selama 24 jam dalam lima minggu. Penelitian akan data set ini dilakukan oleh The Cyber System and Technology Group dari MOT Lincoln Laboratory.

DARPA 1999 memiliki banyak contoh srangan. Data set ini merupakan dalah satu yang *dump file* dengan jenis serangan terlengkap sehingga data set ini sering difunakan untuk penelitian khususnya penelitian yang berhubungan dengan IDS (*Intrusion Detection System*). Beberapa publikasi ilmiah yang menggunakan data set ini antara lain:

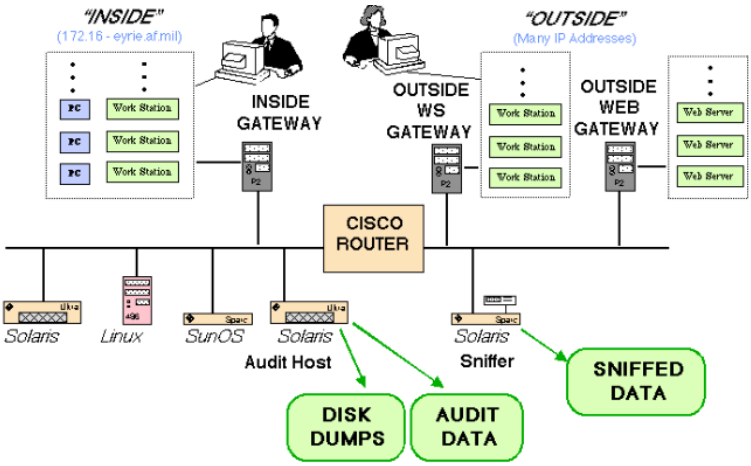
a. Richard P. Lippmann, Robert K. Cunningham, David J. Fried, Issac Graf, Kris R. Kendala, Seth E. Webster, Marc A. Zissman, "Results of the DARPA 1998 Offline Intrusion Detection Evaluation", dipresentasikan di *RAID 1999 Conference,* September 7-9, 1999, West Lafayette, Indiana.
b. Richard P. Lippmann and Robert K. Cunningham, "Using Key-String Selection and Neural Networks to Reduce False Alarms and Detect New Attacks with Sniffer-Based Intrusion DetectionSystem", dipresentasikan di *RAID 1999 Conference,* September 7-9, 1999, West Lafayette, Indiana.
c. R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogord, M. A. Zissman, "Evaluating Intrusion Detection System without Attacking your Friends: The DARPA 1998 Offline Intrusion Detection Evaluation", *SANS 1999*.

Contoh publikasi ilmiah diatas adalah bebrapa publikasi ilmiah yang dihasilkan oleh The Cyber Systems and Technology Group dari MIT Lincoln Laboratory, tentu masih banyak publikasi ilmiah lainnya di luar kelompok tersebut ayng menggunakan data set DARPA 1999.

2.7.1 Arsitektur Simulasi DARPA 1999

DARPA 1999 memiliki arsitektur yang cukup melambangkan aktivitas antar jaringan internal dengan eksternal. Kedua jaringan tersebut dipisahkan oleh sebuah *router* . Jaringan eksternal terdiri dari dua *workstation* yang mensimulasikan

gateway. Menuju jaringan luar secara *virtual*. Setiap *workstation* yang ada pada jaringan eksternal mensimulasikan banyak *virtual workstation* menggunakan perangkat lunak yang disediakan oleh air Force ESC. Jaringan internal meliputi *workstation* yang dijadikan korban, *workstation* ini memiliki sistem operasi bervariasi. Data didapatkan dari salah satu *workstation* yang berada di dalam dan menggunakan perangkat lunak *sniffer* untuk jaringan eksternal. Arsitektur dari DARPA 1999 terdapat pada Gambar 2.4



Gambar 2.4 Arsitektur DARPA 1999

2.7.2 Jenis – jenis Serangan dari DARPA 1999

Subbab ini akan membahas mengenai jenis-jenis serangan yang didapatkan dari uji coba ini selama lima minggu tanpa berhenti. Kategori serangan yang didapatkan terdapat pada Tabel

Jenis-jenis serangan yang didapatkan sangat banyak. Terdapat 43 jenis serangan. Serangan – serangan tersebut dapat dikategorikan menjadi 4 jenis kategori.

DoS (*Denial of Service*) adalah serangan yang bertujuan untuk mencegah server untuk melakukan pelayanan terhadap penggunanya. Pencegahan yang dimaksud adalah mengurangi

peforma hingga mematikan secara total layanan yang disediakan oleh *server*. Cara kerja DoS secara umum adalah membuat aplikasi pada *server crash,* merusak data atau membuat beban kerja dari komputer menjadi banyak. Contoh serangan DoS adalah pemanfaatan *bug* dari aplikasi, menggunakan *bad checksum*, menggunakan *spoofed address,* dan yang paling umum adalah duplikasi paket CP dengan *payload* berbeda.

Kategori selanjutnya adalah *probing*. *Probing* bertujuan untuk menemukan celah dari sistem. Cara menemukan celah tersebut adalah koneksi ke sistem secara tidak penuh missal Nmap dengan tipe *stealth scan* mengirimkan paket TCP tunggal tanpa *handshaking.*

Dua kategori terakhir adalah R2L (remote to local) dan U2R (*User to Root*). R2L adalah usaha untuk melakukan akses sebagai *rootuser* dari koneksi yang dilakukan dari jarak jauh atau udaha untuk mencari kelemahan dari sistem secara koneksi jarak jauh. U2R adalah usaha untuk mendapatkan akses sebagai *rootuser* dari dalam sistem, dalam hal ini penyerang akan masuk ke salam sistem terlebih dahulu sebagai pengguna biasa.

BAB III
DESAIN DAN PERANCANGAN

Pada bab ini akan dijelaskan mengenai hal-hal berkaitan dengan perancangan sistem yang akan dibuat. Perancangan tersebut mencakup deskripsi umum aplikasi, arsitektur sistem, model fungsional, diagram alir aplikasi serta antarmuka aplikasi.

3.1 Deskripsi Umum Sistem

Aplikasi yang dibuat pada Tugas Akhir ini adalah aplikasi pendeteksi intrusi pada lalu lintas jaringan yang berbasis anomali dengan menggunakan metode *n-gram* dan *Incremental Learning*. Aplikasi ini ditempatkan pada *router* dengan sistem operasi Linux. Aplikasi ini akan bertugas sebagai *sniffer*, dimana akan melakukan penangkapan paket data secara terus menerus. Paket yang ditangkap oleh aplikasi ini hanya paket yang menggunakan protokol TCP dan UDP dengan memanfaatkan *library* Jpcap [3]. Paket yang ditangkap akan di rekonstruksi terlebih dahulu dengan mengelompokkan paket yang memiliki IP asal, Port asal, IP tujuan dan Port tujuan yang sama. Setelah selesai direkonstruksi, selanjutnya adalah menghitung distribusi *byte* karakter pada setiap paket menggunakan metode *n-gram* [4] yang kemudian hasil dari proses tersebut akan ditampung kedalam *array*.

Paket-paket yang telah melewati proses perhitungan *n-gram* selanjutnnya adalah proses menghitung jarak mahalanobis paket. Menghitung jarak mahalanobis paket menggunakan metode mahalanobis *distance* [5] antara paket dengan model yang ada. Model yang ada merupakan hasil dari pengolahan paket-paket yang berasal dari DARPA IDS Data Set [7].

Hasil dari perhitungan tersebut akan menghasilkan sebuah nilai. Nilai tersebut akan dibandingkan dengan *threshold* yang sudah ditentukan sebelumnya. Jika nilai tersebut lebih besar dari *threshold* yang ditentukan, maka paket tersebut dapat dikategorikan sebagai paket yang tidak normal. Hasil dari

perhitungan ini kemudian disimpan dalam sebuah *file log* yang akan dibedakan filenya setiap jam.

3.2 Perancangan

Subbab perancangan akan membahas garis besar dan detail dari sistem yang dibangun. Garis besar dan detail dari sistem akan dijelaskan menggunakan diagram alir untuk mempermudah dalam memahami alur kerja sistem.

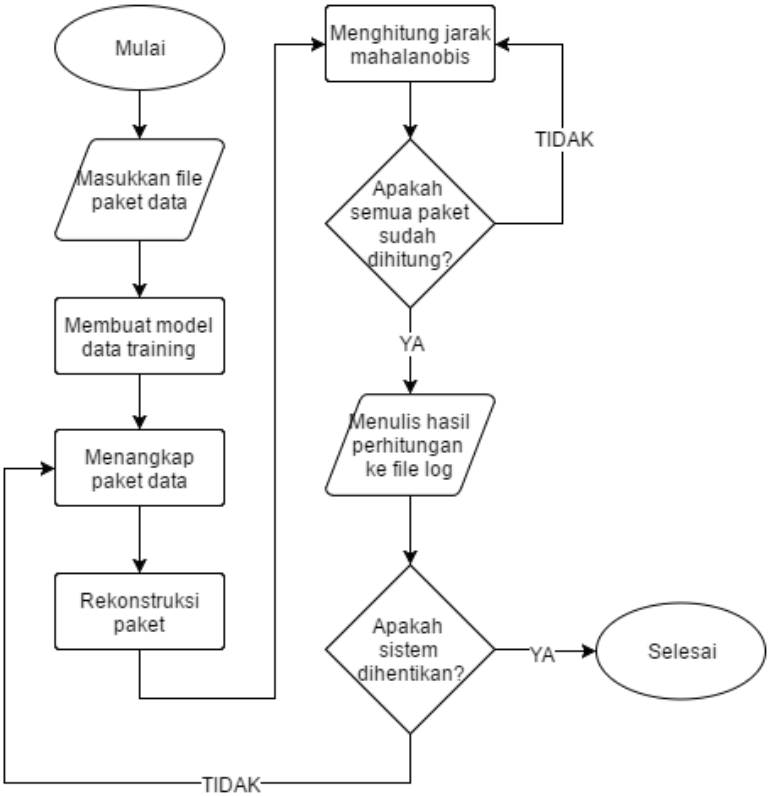
3.2.1 Alur Kerja Sistem Secara Umum

Secara umum sistem yang dibangun terdiri dari tiga proses utama, yaitu proses *training* data set, proses *sniffing* dan proses identifikasi serangan. Alur kerja sistem secara umum dapat dilihat pada Gambar 3.1.

Cara kerja sistem yang lebih detail yaitu, membaca *file* data set, membuat model data *training*, menyimpan model data *training*, menangkap paket, memproses paket dan membandingkan jarak mahalanobis paket dengan model serta pengambilan keputusan terhadap hasil perbandingan. Membuat model data *training* adalah proses dimana membaca *file* paket data dan ditampung pada *array of object*. *File* yang dapat dibaca hanya *file* yang berekstensi **.cap, *.pcap, *.tcpdump* dengan memanfaatkan *library* Jpcap. Proses selanjutnya adalah menangkap *packet* dari *network* interface dengan bantuan library Jpcap lalu menyimpannya pada *array of object*. Selanjutnya adalah proses membandingkan jarak mahalanobis *packet* dengan model data. *Packet* yang dihitung hanya *packet* yang memiliki port tujuan yang kurang dari 1024. Setelah terdapat *packet* yang memenuhi syarat tersebut akan dilakukan proses perhitungan, mulai dari perhitungan rata-rata dan standar deviasi dari *packet*. Dan selanjutnya proses pemanggilan fungsi Mahalanobis *Distance* untuk menghitung jarak mahalanobis *packet* dengan model data *training*. Setelah mendapatkan jarak mahalanobis, lalu dibandingkan dengan nilai *threshold* yang sudah ditentukan sebelumnya. Nilai *threshold*

setiap port memiliki besaran yang berbeda. Jika jarak mahalanobis *packet* melebihi nilai *threshold*, maka *packet* tersebut dapat dikategorikan paket yang tidak normal.

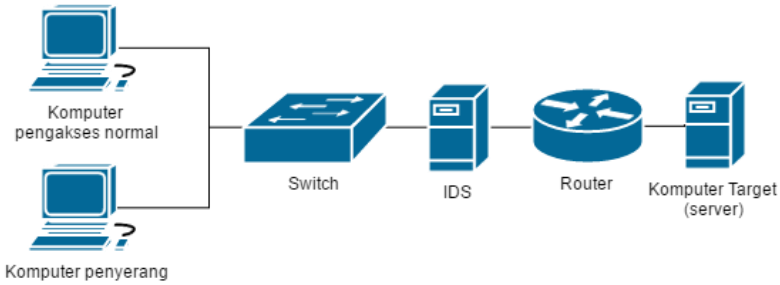
Proses diatas diulangi sampai aplikasi dihentikan oleh pengguna dan menulis hasil keputusan terhadap proses perbandingan ke sebuah *file log*.



Gambar 3.1 Diagram Alir kerja sistem secara umum

3.2.2 Perancangan Arsitektur Jaringan

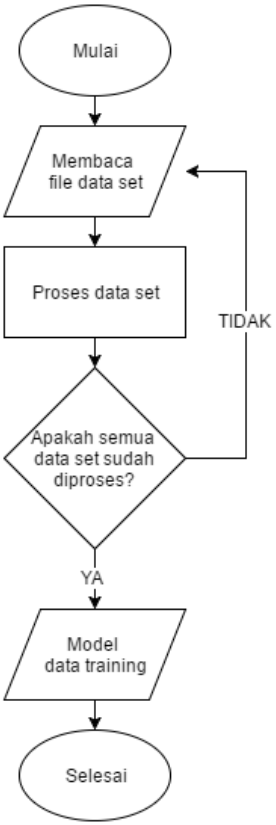
Sistem pendeteksi ini bekerja pada suatu arsitektur jaringan dengan multi subnet yang diilustrasikan pada Gambar 3.2. IDS akan menangkap paket data yang datang maupun keluar dari internet. IDS bekerja sesuai dengan konfigurasi yang dilakukan oleh pengguna. Konfigurasi tersebut disimpan dalam sebuah *file text*.



Gambar 3.2 Topologi jaringan yang akan digunakan

3.2.3 Perancangan Proses *Training* Data Set

Pada bagian ini akan dijelaskan cara program membuat model data *training*. Proses *training* data set merupakan proses yang pertama kali harus dilakukan ketika aplikasi di jalankan. Proses ini merupakan proses yang terpenting pada palikasi yang akan dibangun, karena data set merupakan data yang digunakan sebagai data pembanding dengan paket data yang baru. Pada proses ini aplikasi akan menyimpan model-model paket data normal. Data set yang digunakan merupakan data set dari DARPA tahun 1999 yang merupakan kumpulan paket-paket yang didapat dari kegiatan simulasi pada sebuah arsitektur jaringan yang di desain oleh DARPA sendiri. Untuk alur proses dapat dilihat pada Gambar 3.3.

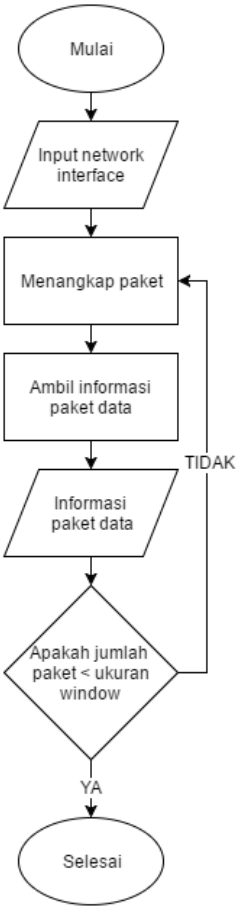


Gambar 3.3 Proses training data set

3.2.4 Perancangan Proses *Sniffing*

Pada bagian ini akan dijelaskan cara program melakukan proses *sniffing*. Proses *sniffing* baru dapat dilakukan ketika sudah ada model data *training*. Jika model data *training* masih kosong sistem akan meminta pengguna untuk menjalankan *training* data set terlebih dahulu. Sniffer ini akan dijalankan pada komputer yang berfungsi sebagai router. Sniffer hanya menangkap paket data yang menggunakan protokol TCP dan UDP, selain itu paket data tidak ditangkap. Sebelum proses *sniffing* berjalan pengguna akan

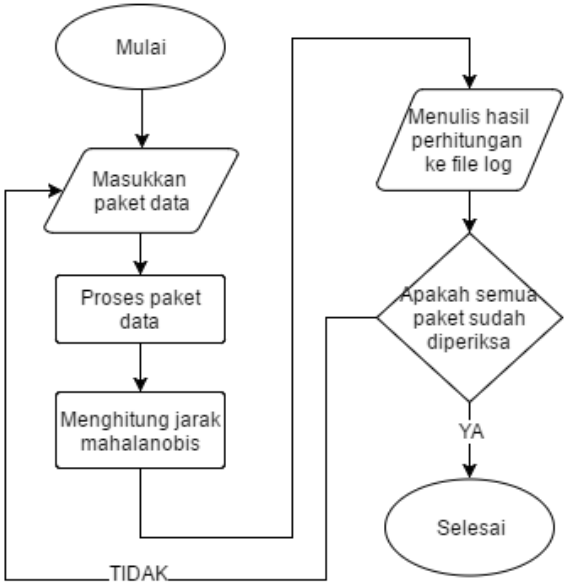
memilih network interface yang akan dimonitor. Selajutnya sistem akan membaca konfigurasi yang terdapat pada file *conf*. Pada file konfigurasi terdapat jumlah paket data yang akan ditangkap oleh sniffer. Jika sniffer sudah menangkap paket data sejumlah yang ditentukan pada file konfigurasi, sniffer akan berhenti menangkap paket lalu mengolah paket data tersebut.



Gambar 3.4 Proses *sniffing*

3.2.5 Perancangan Proses Identifikasi Intrusi

Pada bagian ini akan dijelaskan cara program melakukan deteksi intrusi. Proses deteksi dilakukan setelah proses *training* data set dan *sniffing* selesai. Didalam proses deteksi terdapat sejumlah proses perhitungan data hasil *sniffing* yang harus dilakukan sebelum akhirnya dapat menghasilkan sebuah laporan.



Gambar 3.5 Proses Identifikasi Intrusi

Pada Gambar 3.5 menjabarkan mengenai alur kerja proses deteksi intrusi. Pertama akan dijalankan pemeriksaan apakah Port tujuan paket sama dengan Port tujuan data *training*. Lalu mengambil rat-rata dan standar deviasi yang ada pada model sesuai dengan port paket. Proses selajutnya adalah pemanggilan fungsi Mahalanobis *Distance* dengan mengirimkan parameter *n-gram* dari paket, *n-gram* dari data *training*, standar deviasi dari data *training* dan nilai *smoothing factor*. Proses selanjutnya adalah membandingkan nilai Mahalanobis *Distance* dengan nilai

threshold yang telah ditentukan sebelumnya. Jika nilai Mahalanobis *Distance* dari paket dan data set tersebut lebih besar dari nilai *threshold* maka paket tersebut dapat dikatakan sebagai sebuah intrusi. Lalu program akan menulis informasi paket ke sebuah file log.

3.2.6 Rancangan Antarmuka

Sistem pendeteksi intrusi berbasis anomali ini merupakan sistem yang bekerja dibelakang layar. Hal ini menyebabkan tidak ada antarmuka yang digunakan. Untuk menjembatani konfigurasi sistem yang dilakukan oleh pengguna, maka dibuatlah sebuah *filetext* seperti Gambar 3.6 untuk konfigurasi umum yang biasanya akan dilakukan pengguna.

```
  GNU nano 2.5.3                File: conf

Data Training : /home/ta/dataset/inside/
Data Testing : /home/ta/datates/outside/5/monday/
Threshold : 21-53.30: 23-25.79: 25-219.49: 80-0.0: 53-128.52
Training Status : 0
Smoothing Factor : 0.001
Window Size : 10000
Sniffer Status : 1
```

Gambar 3.6 Contoh *file* konfigurasi

3.2.7 Rancangan Luaran Sistem

Ketika sistem selesai mengolah data hasil *sniffing*, maka akan dihasilkan luaran berupa *log* yang didalamnya terdapat data mengenai olahan data dan identifikasi dari data *sniffing*. Pada Gambar 3.7 ditunjukkan mengenai elemen-elemen yang ada pada *log* nantinya.

```
++++++++++++++++++++++++++++++++++++++++++++
#  Start time : waktu pada saat dijalankan  #
++++++++++++++++++++++++++++++++++++++++++++
Protokol | Date | Source | Destination | Keterangan
--------------------------------------------------
Konten paket data yang berupa serangan
```

Gambar 3.7 Contoh log hasil luaran sistem

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi perancangan perangkat lunak dari aplikasi yang merupakan penerapan data, kebutuhan dan alur sistem yang mengacu pada desain dan perancangan yang telah dibahas sebelumnya. Selain itu, bab ini juga membahas lingkungan pembangunan perangkat lunak yang menjelaskan spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pembangunan sistem.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan dibagi menjadi dua bagian, meliputi perangkat lunak dan perangkat keras.

4.1.1 Perangkat Lunak

Lingkungan implementasi dan pengembangan dilakuakn menggunakan perangkat lunak sebagai berikut:

- Sistem Operasi Linux Ubuntu Desktop 16.04 LTS 64-bit sebagai lingkungan pengembangan sistem secara keseluruhan
- Netbeans IDE 8.1 sebagai IDE utama pembangunan dan pengembangan sistem
- Oracle Java Development Kit (JDK) 1.8 (64-bit) untuk aplikasi penangkapan dan pengolahan paket data
- Jpcap 0.7 untuk library penangkapan dan pengolahan paket data

4.1.2 Perangkat Keras

Lingkungan perangkat keras yang digunakan selama proses pengerjaan Tugas Akhir adalah sebgai berikut:

- Laptop dengan *processor* Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz 2.40GHz, *Installed Memory* (RAM) 8.00 GB dan sistem operasi Linux Ubuntu Desktop 16.04 LTS untuk pengembangan sistem.
- Komputer Aspire M3970 dengan *processor* Intel(R) Core(TM) i3-21202.40GHz 2.40GHz, *Installed Memory* (RAM) 8 GB dan sistem operasi Linux Ubuntu Desktop 16.04 LTS untuk uji coba sistem.

4.2 Implementasi Proses

Berikut adalah implementasi proses deteksi intrusi. Proses-proses akan dijelaskan sesuai dengan urutan berjalannya sistem. Proses dimulai dengan membuat model data *training*, membaca *packet*, memproses *packet*, dan membandingkan jarak mahalanobis *packet* dengan model data *training* yang menghasilkan keputusan terhadap hasil perbandingan. Kemudian hasil perbandingan dan informasi *packet* yang berupa intrusi ditulis ke sebuah file *log*.

4.2.1 Data set

Pada tugas akhir ini penulis menggunakan data set dari 1999 DARPA IDS Data Set sebagai data *training*. Data set yang digunakan terdiri dari beberapa *file* paket data yang berisi sejumlah paket-paket dan mempunyai ukuran *file* yang berbeda serta jumlah *packet* yang berbeda untuk tiap *file*. Data set yang digunakan sebagai data *training* adalah *file* paket data hasil tangkapan paket pada jaringan internal dari DARPA pada minggu pertama dan minggu ke-3. Pada minggu pertama terdapat lima *file* paket data dan minggu ke-3 terdapt tujuh *file* paket data. Keterangan mengenai *file* data set dapat dilihat pada Tabel 4.1.

Tabel 4.1 Data set file Paket Data

No.	Waktu Penagkapan Paket		Nama *File*	Ukuran *File*
	Minggu ke	Hari		
1	1	Senin	inside.tcpdump	325MB
2	1	Selasa	inside.tcpdump	325MB
3	1	Rabu	inside.tcpdump	367MB
4	1	Kamis	inside.tcpdump	527MB
5	1	Jumat	inside.tcpdump	294MB
6	3	Senin	inside.tcpdump	446MB
7	3	Selasa	inside.tcpdump	395MB
8	3	Rabu	inside.tcpdump	533MB
9	3	Kamis	inside.tcpdump	248MB
10	3	Jumat	inside.tcpdump	489MB
11	3	Senin	inside_extra.tcpdump	223MB
12	3	Selasa	inside_extra.tcpdump	438MB
13	3	Rabu	inside_extra.tcpdump	831MB

4.2.2 Implementasi Proses Rekonstruksi Paket Data

Rekonstruksi paket data adalah proses pengelompokan paket data agar dapat digunakan pada proses selanjutnya. Proses rekonstruksi ini bertujuan untuk memisahkan paket yang berupa *request* dari klien dan paket yang berupa *response* dari *server*. Untuk melakukan proses rekonstruksi dibutuhkan beberapa bagian dari paket, bagian yang dibutuhkan dapat dilihat pada Tabel 4.2.

Setelah informasi-informasi didapatkan, program akan memulai proses rekonstruksi. Rekonstruksi yang dimaksud adalah mencocokkan informasi setiap paket yang ada pada *file* paket data. *Pseudocode* untuk rekonstruksi paket data dapat dilihat pada Gambar 4.1.

Tabel 4.2 Daftar bagian paket yang dibutuhkan program

No.	Bagian-bagian paket yang dibutuhkan program
1	Protokol paket
2	IP asal paket
3	Port asal paket
4	IP tujuan paket
5	Port tujuan paket

```
1   Deklarasi ArrayList<DataPacket> datasetTcp
2   Deklarasi ArrayList<DataPacket> dasetUdp
3   Terima paket menggunakan fungsi getPacket()
5   Jika paket termasuk TCP
6       Cocokan IP asal, Port asal, IP tujuan dan
        Port tujuan
7       Tambahkan paket ke datasetTcp
8   Jika paket termasuk UDP
9       Cocokan IP asal, Port asal, IP tujuan dan
        Port tujuan
10      Tambahkan paket ke datasetUdp
```

Gambar 4.1 *Pseudocode* untuk Rekonstruksi paket data

4.2.3 Implementasi Proses Penggunaan Metode *N-Gram*

Implementasi proses pengunaan metode *n-gram* digunakan untuk mengetahui distribusi *byte* karakter pada sebuah paket. Pada tugas akhir ini metode n-gram yang digunakan adalah 1-gram, jadi menghitung kemunculan setiap *byte* karakter sampai *byte* paket habis. *Pseudocode* untuk mengitung N-Gram dapat dilihat pada Gambar 4.2.

```
1   Terima konten paket menggunakan fungsi Ngram()
2   Deklarasi double[] n = new doublbe[256];
3   Baca konten paket
4   Konversi konten paket data menjadi unsign
    integer
5   Tambahkan 1 ke n setiap konten paket yang sesuai
```

Gambar 4.2 *Pseudocode* untuk menghitung N-Gram paket data

4.2.4 Implementasi Perancangan Model Data *Training*

Perancangan model data *training* merupakan proses pembuatan model data *training* untuk sistem sebelum digunakan untuk mendeteksi intrusi. Pada tugas akhir ini model data *training* yang dibuat adalah berdasarkan port tujuan dari sebuah paket data. Model data *training* ini yang nantinya digunakan untuk menghitung jarak mahalanobis setiap paket yang memiliki port tujuan yang sesuai dengan model. *Pseudocode* untuk perancangan mode data *training* dapat dilihat pada Gambar 4.3.

```
1   Deklarasi ArrayList<Double[]> dataTraining
2   Deklarasi ArrayList<DataPacket> datasetTcp
3   Deklarasi ArrayList<DataPacket> dasetUdp
4   Deklarasi ArrayList<DataModel> modelTcp
5   Deklarasi ArrayList<DataModel> modelUdp
6   Periksa tipe protocol dan port tujuan model
7      Jika tipe protocol adalah TCP
8         Baca datasetTcp
9         Jika port datasetTcp sama dengan port
          tujuan model
10           Tambahkan paket ke dataTraining
11        Hitung jumlah variable setiap model
12        Hitung rata-rata variable setiap model
13        Hitung standar variable deviasi setiap
          model
14        Tambahkan model ke modelTcp
15     Jika tipe protocol adalah UDP
16        Baca datasetUdp
17        Jika port datasetUdp sama dengan port
          tujuan model
18           Tambahkan paket ke dataTraining
19        Hitung jumlah variable setiap model
20        Hitung rata-rata variable setiap model
21        Hitung standar deviasi variable setiap
          model
22        Tambahkan model ke modelUdp
```

Gambar 4.3 *Pseudocode* untuk membuat model data *training*

4.2.5 Implementasi *Sniffer*

Fungsi sniffer digunakan untuk menangkap paket data yang melewati host tempat aplikasi ini dijalankan. Paket data yang ditangkap, yaitu paket yang mengunakan protokol TCP atau UDP. Paket yang mengunakan selain protokol TCP atau UDP tidak ditangkap atau dibiarkan lewat. Jumlah paket data yang ditangkap sesuai dengan keinginan pengguna yang dideklarasikan pada *file* konfigurasi. Jika jumlah paket data yang ditangkap belum sesuai dengan yang ditentukan oleh pengguna, maka proses penangkapan paket akan terus dilakukan sampai jumlah paket yang ditangkap sesuai dengan jumlah yang ditentukan sebelumnya. *Pseudocode* untuk *sniffer* dapt dilihat pada Gambar 4.4.

```
1   Deklarasi JpcapCaptor captor
2   Deklarasi NetworkInterface device
3   Deklarasi ukuranWindow
4   Dapatkan masukan device
5   Dapatkan masukan ukuranWindow
6   Deklarasi ArrayList<DataPacket> datasetTcp
7   Deklarasi ArrayList<DataPacket> dasetUdp
8   Tangkap paket menggunakan captor yang ada pada
    device
9      Jika jumlah paket sama dengan ukuranWindow
10        Berhenti tangkap paket
11     Jika tipe protocol paket adalah TCP
12        Rekonstruksi paket
13        Tambahkan paket ke datasetTcp
14     Jika tipe protocol paket adalah UDP
15        Rekonstruksi paket
16        Tambahkan paket ke datasetUdp
```

Gambar 4.4 *Pseudocode* untuk *sniffer*

4.2.6 Implementasi Proses Penggunaan Metode Mahalanobis *Distance*

Implementasi proses penggunaan metode mahalanobis *distance* digunakan untuk menghitung jarak mahalanobis antara paket baru dan model data *training*. *Pseudocode* untuk menghitung

jarak mahalanobis menggunakan Mahalanobis *Distance* dapat dilihat pada Gambar 4.5.

```
1   Deklarasi ngram
2   Deklarasi mean
3   Deklarasi standarDeviasi
4   Deklarasi smoothingFactor
5   Deklarasi jarak
6   jarak = nilai mutlak dari (ngram-mean) dibagi
    (standarDeviasi+smoothingFactor)
```

Gambar 4.5 *Pseudocode* penggunaan metode Mahalanobis Distance

4.2.7 Implementasi Pendeteksian Intrusi

Implementasi pendeteksian intrusi merupakan proses menentukan apakah paket data yang diperiksa merupakan paket data normal atau paket data yang berupa intrusi. Proses ini dilakukan dengan membandingkan nilai jarak mahalanobis paket dengan nilai *threshold* yang sudah ditentukan sebelumnya. Jika nilai jarak mahalanobis paket lebih besar dari *threshold* maka paket tersebut dapat dikatakan sebagai paket tidak normal. Pseudocode untuk pendeteksian intrusi dapat dilihat pada Gambar 4.6

```
1   Deklarasi jarak
2   Deklarasi threshold
3   Jika nilai jarak lebih besar dari nilai
    threshold
4       Paket tersebut dianggap intrusi
5       Tulis informasi paket ke file log
```

Gambar 4.6 *Pseudocode* untuk Pendeteksian Intrusi

4.2.8 Implementasi Proses *Incremental Learning*

Implementasi proses *incremental learning* digunakan untuk memperbaharui model data *training*. Nilai yang diperbaharui dari model, yitu rata-rata dan standar deviasi dari model yang sesuai. *Pseudocode* untuk proses *incremental learning* dapat dilihat pada Gambar 4.7.

```
1   Deklarasi ngram
2   Deklarasi modelTcp
3   Deklarasi modelUdp
4   Deklarasi tipeProtoko
5   Deklarasi portPaket
6   Deklarasi mean
7   Deklarasi standarDeviasi
8   Dapatkan masukan tipeProtocol
9   Dapatkan masukan portPaket
10  Dapatkan masukan ngram
11  Jika tipe protocol adalah TCP
12     Ambil mean dari modelTcp
13     Ambil standarDeviasi dari modelTcp
14     Hitung mean yang baru dengan ngram
15     Hitung standarDeviasi yang baru dengan ngram
16     Perbaharui mean pada modelTcp sesuai
       portPaket
17     Perbaharui standarDeviasi modelTcp sesuai
       portPaket
18  Jika tipe protocol adalah TCP
19     Ambil mean dari modelTcp
20     Ambil standarDeviasi dari modelTcp
21     Hitung mean yang baru dengan ngram
22     Hitung standarDeviasi yang baru dengan ngram
23     Perbaharui mean pada modelTcp sesuai
       portPaket
24     Perbaharui standarDeviasi modelTcp sesuai
       portPaket
```

Gambar 4.7 *Pseudocode* untuk *Incremental Learning*

LAMPIRAN

Bagian ini merupakan lapiran sebagai dokumen pelengkap dari buku Tugas Akhir, pada bagian ini diberikan informasi mengenai kode sumber dari sistem yang dibuat.

A. Kode Sumber

A.1. Kode Sumber Proses Rekonstruksi Paket Data

```
import java.nio.charset.StandardCharsets;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.TimeZone;
import jpcap.JpcapCaptor;
import jpcap.packet.Packet;
import jpcap.packet.TCPPacket;
import jpcap.packet.UDPPacket;
/**
 *
 * @author agus
 */
```

```
public class PacketReader implements Runnable {
    private static int countPacket;
    private int input, files, type, counter, windowSize;
    private double[] numChars;
    private String tuples, timeFormat, startTime, regex = "\\r?\\n";
    private String[] header, time, line, split;
    private byte[] data;
    private Date date;
    private DateFormat format;
    private JpcapCaptor captor;
    private TCPPacket tcp;
    private UDPPacket udp;
    private ArrayList<DataPacket> datasetTcp;
    private ArrayList<DataPacket> datasetUdp;
    private ArrayList<DataPacket> dataTest;
    private Map<String, BodyPacket> packetBody = new HashMap<>();
    private Map<String, String> packetTime = new HashMap<>();
    private Map<String, String> dataPort;
    Ngram ng = new Ngram();
    BodyPacket bp;

    public PacketReader(){

    }

    public PacketReader(int files, JpcapCaptor captor, int input,
ArrayList<DataPacket> datasetTcp, ArrayList<DataPacket> datasetUdp,
```

```
ArrayList<DataPacket> dataTest, Map<String, String> dataPort, int type, int
windowSize){
        this.files = files;
        this.captor = captor;
        this.input = input;
        this.datasetTcp = datasetTcp;
        this.datasetUdp = datasetUdp;
        this.dataTest = dataTest;
        this.dataPort = dataPort;
        this.type = type;
        this.windowSize = windowSize;
    }

    @Override
    public void run(){
        while (true) {
            Packet packet = captor.getPacket();

            synchronized(packetBody){
                if (packet == null || packet == Packet.EOF || (input == 3 &&
counter == windowSize)) break;

                if (packet instanceof TCPPacket && packet.data.length != 0){
                    tcp = (TCPPacket) packet;
                    if (dataPort.containsKey("TCP"+tcp.dst_port)) {
                        if (input == 3) {
                            time = new String(tcp.toString()).split(":");
                            date = new Date(Long.parseLong(time[0])*1000L);
```

```
                            format = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");

format.setTimeZone(TimeZone.getTimeZone("America/New_York"));
                            timeFormat = format.format(date);
                        }
                        tuples = input+"-TCP-
"+tcp.src_ip.toString().substring(1)+"-"+tcp.src_port+"-
"+tcp.dst_ip.toString().substring(1)+"-"+tcp.dst_port;
                        if (packetBody.containsKey(tuples)){
                            bp = packetBody.get(tuples);
                            bp.addBytes(tcp.data);
                        } else {
                            bp = new BodyPacket(tcp.data);
                            packetBody.put(tuples, bp);
                            packetTime.put("TCP-
"+tcp.src_ip.toString().substring(1)+"-"+tcp.src_port+"-
"+tcp.dst_ip.toString().substring(1)+"-"+tcp.dst_port, timeFormat);
                        }
                        countPacket++;
                        counter++;
                    }
                }
                else if(packet instanceof UDPPacket && packet.data.length != 0){
                    udp = (UDPPacket) packet;
                    if (dataPort.containsKey("UDP"+udp.dst_port)) {
                        if (input == 3) {
                            time = new String(udp.toString()).split(":");
                            date = new Date(Long.parseLong(time[0])*1000L);
```

```
                            format = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");

format.setTimeZone(TimeZone.getTimeZone("America/New_York"));
                            timeFormat = format.format(date);
                        }
                        tuples = input+"-UDP-
"+udp.src_ip.toString().substring(1)+"-"+udp.src_port+"-
"+udp.dst_ip.toString().substring(1)+"-"+udp.dst_port;
                        if(packetBody.containsKey(tuples)){
                            bp = packetBody.get(tuples);
                            bp.addBytes(udp.data);
                        } else {
                            bp = new BodyPacket(udp.data);
                            packetBody.put(tuples, bp);
                            packetTime.put("UDP-
"+udp.src_ip.toString().substring(1)+"-"+udp.src_port+"-
"+udp.dst_ip.toString().substring(1)+"-"+udp.dst_port, timeFormat);
                        }
                        countPacket++;
                        counter++;
                    }
                }
            }
        }
        for (Map.Entry<String, BodyPacket> entry : packetBody.entrySet()) {
            String key = entry.getKey();
            header = key.split("-", 0);
            BodyPacket value = entry.getValue();
```

```
            numChars = ng.Ngram(value.getBytes());
            if (header[0].equals("1") && header[1].equals("TCP")) {
                datasetTcp.add(new DataPacket(startTime, header[1], header[2],
Integer.parseInt(header[3]), header[4], Integer.parseInt(header[5]), null,
numChars, type));
            }
            else if (header[0].equals("1") && header[1].equals("UDP")) {
                datasetUdp.add(new DataPacket(startTime, header[1], header[2],
Integer.parseInt(header[3]), header[4], Integer.parseInt(header[5]), null,
numChars, type));
            }
            else {
                startTime = packetTime.get(header[1]+"-"+header[2]+"-
"+header[3]+"-"+header[4]+"-"+header[5]);
                dataTest.add(new DataPacket(startTime, header[1], header[2],
Integer.parseInt(header[3]), header[4], Integer.parseInt(header[5]),
value.getBytes(), numChars, type));
            }
        }
        packetBody.clear();
        packetTime.clear();
    }
}
```

A.2. Kode Sumber Proses Penggunaan Metodel N-Gram

```
package ids;

/**
 *
 * @author agus
 */
public class Ngram {
    private int ascii;
    private double[] n;

    public double[] Ngram(byte[] data){
        if (data != null) {
            n = new double[256];

            for (byte b : data) {
                ascii = b & 0xFF;
                n[ascii] += 1;
            }
        }
        return n;
    }
}
```

A.3. Kode Sumber Proses Perancangan Model Data Training

```
package ids;

import java.util.ArrayList;
import org.apache.commons.lang3.ArrayUtils;

/**
 *
 * @author agus
 */
public class DataTraining implements Runnable {
    private int port, ascii = 256;
    private String proto;
    private double[] sumData = new double[ascii], meanData = new double[ascii],
deviasiData = new double[ascii], quadraticData = new double[ascii], standardData
= new double[ascii];
    private ArrayList<DataPacket> datasetTcp;
    private ArrayList<DataPacket> datasetUdp;
    private ArrayList<DataModel> modelTcp;
    private ArrayList<DataModel> modelUdp;
    private ArrayList<Double[]> dataTraining = new ArrayList<>();

    public DataTraining(String proto, ArrayList<DataPacket> datasetTcp,
ArrayList<DataPacket> datasetUdp, ArrayList<DataModel> modelTcp,
ArrayList<DataModel> modelUdp, int port){
        this.proto = proto;
```

```
        this.datasetTcp = datasetTcp;
        this.datasetUdp = datasetUdp;
        this.modelTcp = modelTcp;
        this.modelUdp = modelUdp;
        this.port = port;
    }

    public void run(){
        synchronized(dataTraining){
            if (proto.equals("TCP")) {
                for (DataPacket dataSetTcp : datasetTcp) {
                    if (dataSetTcp.getDstPort() == port) {

dataTraining.add(ArrayUtils.toObject(dataSetTcp.getNgram()));
                    }
                }

                for (int i = 0; i < dataTraining.size(); i++) {
                    for (int j = 0; j < ascii; j++) {
                        sumData[j] += dataTraining.get(i)[j];
                        quadraticData[j] += Math.pow(dataTraining.get(i)[j], 2);
                    }
                }

                for (int i = 0; i < meanData.length; i++) {
                    meanData[i] = sumData[i]/dataTraining.size();
                }
```

```
                for (int i = 0; i < deviasiData.length; i++) {
                    deviasiData[i] =
Math.sqrt((dataTraining.size()*quadraticData[i]-Math.pow(sumData[i],
2))/(dataTraining.size()*(dataTraining.size()-1)));
                }

                modelTcp.add(new DataModel(port, sumData, meanData, deviasiData,
quadraticData, dataTraining.size()));
            }

            else if (proto.equals("UDP")) {
                for (DataPacket dataSetUdp : datasetUdp) {
                    if (dataSetUdp.getDstPort() == port) {

dataTraining.add(ArrayUtils.toObject(dataSetUdp.getNgram()));
                    }
                }

                for (int i = 0; i < dataTraining.size(); i++) {
                    for (int j = 0; j < ascii; j++) {
                        sumData[j] += dataTraining.get(i)[j];
                        quadraticData[j] += Math.pow(dataTraining.get(i)[j], 2);
                    }
                }

                for (int i = 0; i < meanData.length; i++) {
                    meanData[i] = sumData[i]/dataTraining.size();
                }
```

```
                for (int i = 0; i < deviasiData.length; i++) {
                    deviasiData[i] =
Math.sqrt((dataTraining.size()*quadraticData[i]-Math.pow(sumData[i],
2))/(dataTraining.size()*(dataTraining.size()-1)));
                }

                modelUdp.add(new DataModel(port, sumData, meanData, deviasiData,
quadraticData, dataTraining.size()));
            }
        }
    }

}
```

A.4. Kode Sumber Sniffing

```
NetworkInterface[] device = JpcapCaptor.getDeviceList();
for (int i = 0; i < device.length; i++) {
	System.out.println(i+": "+device[i].name + "(" +
device[i].description+")");
	//print out its datalink name and description
	System.out.println("   Datalink: "+device[i].datalink_name + "(" +
device[i].datalink_description+")");
	//print out its MAC address
	System.out.print("   MAC address: ");
	for (byte b : device[i].mac_address)
		System.out.print(Integer.toHexString(b&0xff) + ":");
	System.out.println();
	//print out its IP address, subnet mask and broadcast address
	for (NetworkInterfaceAddress a : device[i].addresses)
		System.out.println("   Address: "+a.address + " " + a.subnet + " "+
a.broadcast);
	System.out.println("\n");
}
System.out.println("Choose active network interface...(0,1,2)!");
sc = new Scanner(System.in);
input = sc.nextInt();
System.out.println("Selected network interface name : "+device[input].name);
fwRunning.append("Selected network interface name : "+device[input].name+"\n");
snifferStatus = true;
while (snifferStatus) {
```

```
        System.out.println("Start Sniffing...");
        time = ids.getDateTime();
        windowSize = Integer.parseInt(ids.getData("Window Size "));
        thresholdAll = ids.getData("Threshold ");
        portTh = ids.getThreshold();
        sFactor = Double.parseDouble(ids.getData("Smoothing Factor "));
        dateTime = time.split("_");
        fileLog = new File(dateTime[0]+"/"+dateTime[1]+"/"+dateTime[2]);
        if (!fileLog.exists()) {
              fileLog.mkdirs();
        }
        fileLog = new
File(dateTime[0]+"/"+dateTime[1]+"/"+dateTime[2]+"/Sniff_Result_log_"+dateTime[3]
);
        fileRecord = new
File(dateTime[0]+"/"+dateTime[1]+"/"+dateTime[2]+"/Sniff_Record_log_"+dateTime[3]
);
        fileRunning = new
File(dateTime[0]+"/"+dateTime[1]+"/"+dateTime[2]+"/Running_Test_log");
        if (!fileLog.exists() | !fileRecord.exists()) {
              fileLog.createNewFile();
              fileRecord.createNewFile();
              fileRunning.createNewFile();
        }
        fwLog = new FileWriter(fileLog,true);
        fwRecord = new FileWriter(fileRecord, true);
        freePacket = new ArrayList<>();
        attackPacket = new ArrayList<>();
```

```
        System.out.println("Window Size : "+windowSize);
        fwRunning.append("Window Size : "+windowSize+"\n");
        PacketSniffer ps = new PacketSniffer(device[input], input, dataTest,
dataPort, windowSize);
        Thread threadPs = new Thread(ps);
        start = System.currentTimeMillis();
        threadPs.start();
        try {
                threadPs.join();
        } catch (InterruptedException ex) {
                Logger.getLogger(IDS.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

A.5. Kode Sumber Proses Penggunaan Metode Mahalanobis Distance

```
package ids;

/**
 *
 * @author agus
 */
public class Mahalanobis {

    public double distance(double[] x, double[] y, double[] sd, double sf){
        double sum = 0;

        for (int i = 0; i < x.length; i++) {
            sum += Math.abs(x[i] - y[i])/(sd[i]+sf);
        }
        return sum;
    }

}
```

A.6. Kode Sumber Proses Pendeteksian Serangan

```
for (DataPacket dataPacketTes : dataTest) {
       if ("TCP".equals(dataPacketTes.getProtokol())) {
              for (DataModel dataTcp : modelTcp) {
                     if (dataTcp.getDstPort() == dataPacketTes.getDstPort()) {
                            mahalanobis = new Mahalanobis();
                            mDist = mahalanobis.distance(dataPacketTes.getNgram(),
dataTcp.getMeanData(), dataTcp.getDeviasiData(),sFactor);
                            if (mDist > portTh[dataPacketTes.getDstPort()]) {
                                   System.out.println("TCP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" |
"+Math.round(mDist*100.0)/100.0+"\n");
                                   fwLog.append("TCP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" |
"+Math.round(mDist*100.0)/100.0+"\n");

       fwLog.append("++++++++++++++++++++++++++++++++++START++++++++++++++++++
++++++++++++++++\n");
                                   fwLog.append(new
String(dataPacketTes.getPacketData(), StandardCharsets.US_ASCII)+"\n");
```

```
       fwLog.append("++++++++++++++++++++++++++++++++++++END++++++++++++++++++
++++++++++++++++\n");
                                    fwRecord.append("TCP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" | Attack\n");
                                    attackPacket.add(Math.round(mDist*100.0)/100.0);
                             } else {
                                    fwRecord.append("TCP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" | Normal\n");
                                    ids.incremental("TCP", dataPacketTes.getNgram(),
dataPacketTes.getDstPort());
                                    freePacket.add(Math.round(mDist*100.0)/100.0);
                             }
                      }
              }
       }

       else if ("UDP".equals(dataPacketTes.getProtokol())) {
              for (DataModel dataUdp : modelUdp) {
                     if (dataUdp.getDstPort() == dataPacketTes.getDstPort()) {
                            mahalanobis = new Mahalanobis();
                            mDist = mahalanobis.distance(dataPacketTes.getNgram(),
dataUdp.getMeanData(), dataUdp.getDeviasiData(),sFactor);
                            if (mDist > portTh[dataPacketTes.getDstPort()]) {
```

```
                                    System.out.println("UDP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" |
"+Math.round(mDist*100.0)/100.0+"\n");
                                    fwLog.append("UDP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" |
"+Math.round(mDist*100.0)/100.0+"\n");

       fwLog.append("++++++++++++++++++++++++++++++++++START++++++++++++++++
++++++++++++++++\n");
                                    fwLog.append(new
String(dataPacketTes.getPacketData(), StandardCharsets.US_ASCII)+"\n");

       fwLog.append("++++++++++++++++++++++++++++++++++END++++++++++++++++++
++++++++++++++++\n");
                                    fwRecord.append("UDP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" | Attack\n");
                                    attackPacket.add(Math.round(mDist*100.0)/100.0);
                               } else {
                                    fwRecord.append("UDP |
"+dataPacketTes.getTimePacket()+" |
"+dataPacketTes.getSrcIP()+":"+dataPacketTes.getSrcPort()+" |
"+dataPacketTes.getDstIP()+":"+dataPacketTes.getDstPort()+" | Normal\n");
```

```
                                    ids.incremental("UDP", dataPacketTes.getNgram(),
dataPacketTes.getDstPort());
                                    freePacket.add(Math.round(mDist*100.0)/100.0);
                                }
                            }
                        }
                    }
}
```

A.7. Kode Sumber Proses Incremental Learning

```
private void incremental(String proto, double[] ngram, int port){
	if (proto.equals("TCP")) {
		for (DataModel dataTcp : modelTcp) {
			if (dataTcp.getDstPort() == port) {
				int numNew = dataTcp.getTotalModel();
				double[] sumNew = dataTcp.getSumData();
				for (int i = 0; i < sumNew.length; i++) {
					sumNew[i] = sumNew[i]+ngram[i];
				}
				double[] quadraticNew = dataTcp.getQuadraticData();
				for (int i = 0; i < quadraticNew.length; i++) {
					quadraticNew[i] =
quadraticNew[i]+Math.pow(ngram[i], 2);
				}
				double[] meanNew = dataTcp.getMeanData();
				for (int i = 0; i < meanNew.length; i++) {
					meanNew[i] =
(meanNew[i]*numNew+ngram[i])/(numNew+1);
				}
				numNew = numNew + 1;
				double[] deviasiNew = dataTcp.getDeviasiData();
				for (int i = 0; i < deviasiNew.length; i++) {
					deviasiNew[i] =
Math.sqrt((numNew*quadraticNew[i]-Math.pow(sumNew[i], 2))/(numNew*(numNew-1)));
				}
```

```
                        dataTcp.setSumData(sumNew);
                        dataTcp.setDeviasiData(deviasiNew);
                        dataTcp.setMeanData(meanNew);
                        dataTcp.setQuadraticData(quadraticNew);
                        dataTcp.setTotalModel(numNew);
                    }
                }
        }

        else if (proto.equals("UDP")) {
                for (DataModel dataUdp : modelUdp) {
                    if (dataUdp.getDstPort() == port) {
                        int numNew = dataUdp.getTotalModel();
                        double[] sumNew = dataUdp.getSumData();
                        for (int i = 0; i < sumNew.length; i++) {
                            sumNew[i] = sumNew[i]+ngram[i];
                        }
                        double[] quadraticNew = dataUdp.getQuadraticData();
                        for (int i = 0; i < quadraticNew.length; i++) {
                            quadraticNew[i] =
quadraticNew[i]+Math.pow(ngram[i], 2);
                        }
                        double[] meanNew = dataUdp.getMeanData();
                        for (int i = 0; i < meanNew.length; i++) {
                            meanNew[i] =
(meanNew[i]*numNew+ngram[i])/(numNew+1);
                        }
                        numNew = numNew + 1;
```

```
                                double[] deviasiNew = dataUdp.getDeviasiData();
                                for (int i = 0; i < deviasiNew.length; i++) {
                                      deviasiNew[i] =
Math.sqrt((numNew*quadraticNew[i]-Math.pow(sumNew[i], 2))/(numNew*(numNew-1)));
                                }
                                dataUdp.setSumData(sumNew);
                                dataUdp.setDeviasiData(deviasiNew);
                                dataUdp.setMeanData(meanNew);
                                dataUdp.setQuadraticData(quadraticNew);
                                dataUdp.setTotalModel(numNew);
                          }
                    }
              }
}
```

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai kesimpulan yang dapat diambil dari perancangan sistem hingga hasil pengujian. Selain itu juga akan dibahas mengenai hasil yang sudah dicapai dan belum dicapai. Pada bab ini juga akan menjawab pertanyaan yang dikemukakan pada Bab 1. Pada penutup ini juga terdapat saran-saran untuk pengembangan selanjutnya.

6.1 Kesimpulan

Dalam proses pengerjaan Tugas Akhir yang melalui tahap perancangan, implementasi, serta uji coba, didapatkan kesimpulan sebagai berikut :

1. Metode Mahalanobis *Distance* tidak dapat digunakan untuk mengklasifikasikan antara paket data normal dan paket data yang berupa intrusi untuk protokol HTTP. Jarak yang dihasilkan pada saat *training* menggunakan paket data normal maupun paket data yang berupa intrusi yaitu bernilai 0. Sehingga paket data normal maupun paket data intrusi tidak dapat dibedakan.
2. Sistem yang dibuat untuk pendeteksi intrusi menggunakan metode Mahalanobis *Distance* tanpa proses *incremental learning* dapat mendeteksi intrusi dengan persentase kebenaran sekitar 93%, namun dengan tambahan proses *incremental learning* hanya dapat mendeteksi intrusi dengan persentase kebenaran sekitar 20%. Dari hasil tersebut, dengan tambahan proses incremental learning mangurangi tingkat akurasi pendeteksian intrusi.

6.2 Saran

Adapun saran-saran yang diberikan untuk pengembangan sistem ini selanjutnya adalah karena membedakan paket data normal dengan paket data serangan menggunakan metode

mahalanobis *distance* dengan proses *incremental learning* kurang akurat dibandingkan tanpa proses *incremental leraning*. Hal ini dikarenakan dengan menambahkan proses *incremental learning*, rata-rata dan standar deviasi pada model diperbaharui tetapi *threshold* yang digunakan untuk mendeteksi intrusi tidak diperbaharui, sehingga *threshold* yang ada tidak akurat untuk mendeteksi intrusi. Perlu ada implementasi metode lain sehingga dapat membantu meningkatkan keakuratan pendeteksian intrusi.

DAFTAR PUSTAKA

[1] SANS Institute, "Understanding Intrusion Detection System," *SANS Institute Reading Room*, pp. 1-9, 2001.

[2] "Intrusion detection system," [Online]. Available: https://en.wikipedia.org/wiki/Intrusion_detection_system. [Diakses 22 June 2016].

[3] K. Fuji, "a Java library for capturing and sending network packets," Jpcap, 15 May 2007. [Online]. Available: http://jpcap.gitspot.com/. [Diakses 23 May 2016].

[4] A. Hanafi, "Pengenalan Bahasa Suku Bangsa Indonesia Berbasis Teks Menggunakan Metode N-gram. IT TELKOM," 2009.

[5] "Mahalanobis distance," [Online]. Available: https://en.wikipedia.org/wiki/Mahalanobis_distance. [Diakses 22 June 2016].

[6] D. E. Knuth, "The Art of Computer Programming," *Fundamental Algorithms. Addison Wesley*, vol. 1, 1973.

[7] MIT Lincoln Laboratory, "MIT Lincoln Laboratory: Cyber system & technolog: DARPA Intrusion Detection," MIT Lincoln Laboratory, [Online]. Available: https://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/docs/index.html. [Diakses 23 Mei 2016].

[8] V. Galleys, "Cross Validation," 2006. [Online]. Available: http://www.cse.iitb.ac.id/~tarung/smt/papers_ppt/ency-cross-validation.pdf. [Diakses 24 June 2016].

[9] Kohavi, "Confusion Matrix," 1999. [Online]. Available: http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matix/confusion_matrix.html. [Diakses 24 June 2016].

BIODATA PENULIS

I Made Agus Adi Wirawan, lahir di Desa Celagi, 4 Agustus 1994. Penulis adalah anak kedua dari dua bersaudara. Menempuh pendidikan di SD No. 3 Denbantas, SMP Negeri 1 Tabanan, SMA Negeri 1 Tabanan, dan terakhir melanjutkan kuliah di jurusan Teknik Informatika – ITS.

Selama berkuliah penulis aktif dalam kegiatan dan organisasi keprofesian informatika sebagai administrator sekaligus koordinator laboratorium Arsitektur dan Jaringan Komputer. Pernah mengikuti dan mendapatkan sertifikasi HCNA-WCDMA yang diselenggarakan oleh Huawei.

Selain itu, penulis juga aktif dalam organisasi kampus sebagai anggota Himpunan Mahasiswa Teknik Computer-Informatika, staf departemen dalam negeri dan menjadi panitia berbagai kegiatan di tingkat jurusan maupun fakultas.

Selain menjalankan tugas mahasiswa, penulis juga aktif menjadi asisten dosen mata kulish sistem operasi, jaringan komputer, dan keamanan informasi dan jaringan, disamping asisten dosen juga sekaligus menjadi asisten praktikum untuk mata kuliah sistem operasi dan jaringan komputer,

Ketertarikan penulis dibidang informatika berada pada bidang sistem teknologi informasi, sekuritas jaringan, perancangan keamanan sistem dan jaringan, teknologi antar jaringan, dan teknologi tepat guna.

Penulis dapat dihubungi dengan mengirimkan pesan elektronik ke alamat imadeagus.04@gmail.com.