



TUGAS AKHIR - KI141502

SISTEM PENDETEKSI SERANGAN 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**



TUGAS AKHIR - KI141502

SISTEM PENDETEKSI SERANGAN 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**

[Halaman ini sengaja dikosongkan]



UNDERGRADUATE THESES - KI141502

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**

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Sistem Pendeteksi Serangan 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
NIP: 197708242026241001

.....
(pembimbing 1)

BASKORO ADI PRATOMO, S.Kom.,
M.Kom
NIP: 198702182014041001

.....
(pembimbing 2)

SURABAYA
JULI, 2016

[Halaman ini sengaja dikosongkan]

SISTEM PENDETEKSI SERANGAN 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 serangan 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 informasi 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

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 which 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 whether 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

KATA PENGANTAR

Puji syukur setinggi-tingginya bagi Ida Sang Hyang Widhi Wasa, yang telah memberikan berkah dan kelancaran sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul “Sistem Pendeteksi Serangan Berbasis Anomali dengan n-gram dan *Incremental Learning*” dengan tepat waktu.

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat berharga bagi penulis, karena dengan mengerjakan Tugas Akhir ini penulis dapat memperdalam, meningkatkan serta mengimplementasikan ilmu yang didapat selama penulis menempuh perkuliahan di jurusan Teknik Informatika ITS.

Terselesaikannya buku Tugas Akhir ini, tidak lepas dari bantuan dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Ida Sang Hyang Widhi Wasa atas berkah yang tiada habisnya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Bapak, Ibu, dan kakak penulis I Putu Andy Sudarmawan yang telah memberikan dukungan moral dan material serta doa yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Bapak Baskoro Adi Pratomo, S.Kom., M.Kom selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi penulis dalam mengerjakan Tugas Akhir ini.
5. Bapak Darlis Herumurti, S.Kom., M.Kom selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom., M.Sc selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Teman-teman Laboratorium AJK, agan Romen, Eva, Surya, Thiar, Pur, Zaza, Wicak, Uul, Asbun, Daniel, Nindi, RI sma,

Syukron, Fatih, dan Oink yang senantiasa menghibur dan mendukung penulis dalam mengerjakan tugas akhir ini serta menemani penulis di laboratorium.

7. Teman-teman TC angkatan 2012 yang sudah bersama-sama jatuh bangun menjalani kuliah di kampus TC sejak maba hingga akhir kuliah.
8. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Sebagai manusia biasa, penulis menyadari Tugas Akhir ini masih jauh dari kesempurnaan dan memiliki banyak kekurangan. Sehingga dengan segala kerendahan hati penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Juli 2016

I Made Agus Adi Wirawan

DAFTAR ISI

| | |
|--|------|
| LEMBAR PENGESAHAN..... | v |
| Abstrak..... | vii |
| Abstract..... | ix |
| KATA PENGANTAR..... | xi |
| DAFTAR ISI..... | xiii |
| DAFTAR GAMBAR..... | xvii |
| DAFTAR TABEL..... | xix |
| DAFTAR PERSAMAAN..... | xxi |
| BAB I PENDAHULUAN..... | 1 |
| 1.1 Latar Belakang..... | 1 |
| 1.2 Rumusan Masalah..... | 2 |
| 1.3 Batasan Masalah..... | 2 |
| 1.4 Tujuan..... | 2 |
| 1.5 Manfaat..... | 3 |
| 1.6 Metodologi..... | 3 |
| 1.7 Sistematika Penulisan Laporan Tugas Akhir..... | 4 |
| BAB II TINJAUAN PUSTAKA..... | 7 |
| 2.1 IDS..... | 7 |
| 2.2 IDS Berbasis Anomali..... | 8 |
| 2.3 Jpcap..... | 8 |
| 2.4 N-Gram..... | 10 |
| 2.5 Simplified Mahalanobis Distance..... | 11 |
| 2.6 <i>Incremental Learning</i> | 13 |
| 2.7 DARPA 1999..... | 14 |
| 2.7.1 Arsitektur Simulasi DARPA 1999..... | 15 |
| 2.7.2 Jenis – jenis Serangan dari DARPA 1999..... | 16 |
| BAB III DESAIN DAN PERANCANGAN..... | 19 |
| 3.1 Deskripsi Umum Sistem..... | 19 |
| 3.2 Perancangan..... | 20 |
| 3.2.1 Alur Kerja Sistem Secara Umum..... | 20 |
| 3.2.2 Perancangan Arsitektur Jaringan..... | 22 |
| 3.2.3 Perancangan Proses <i>Training</i> Data Set..... | 22 |
| 3.2.4 Perancangan Proses <i>Sniffing</i> | 23 |

| | | |
|----------------------------------|--|----|
| 3.2.5 | Perancangan Proses Identifikasi Serangan | 25 |
| 3.2.6 | Rancangan Antarmuka | 26 |
| 3.2.7 | Rancangan Luaran Sistem | 26 |
| BAB IV IMPLEMENTASI..... | | 27 |
| 4.1 | Lingkungan Implementasi | 27 |
| 4.1.1 | Perangkat Lunak..... | 27 |
| 4.1.2 | Perangkat Keras..... | 27 |
| 4.2 | Implementasi Proses..... | 28 |
| 4.2.1 | Data set..... | 28 |
| 4.2.2 | Implementasi Proses Rekonstruksi Paket Data | 29 |
| 4.2.3 | Implementasi Proses Penggunaan Metode <i>N-Gram</i> | 30 |
| 4.2.4 | Implementasi Perancangan Model Data <i>Training</i> | 31 |
| 4.2.5 | Implementasi <i>Sniffer</i> | 32 |
| 4.2.6 | Implementasi Proses Penggunaan Metode Mahalanobis <i>Distance</i> | 32 |
| 4.2.7 | Implementasi Pendeteksian Serangan..... | 33 |
| 4.2.8 | Implementasi Proses <i>Incremental Learning</i> | 33 |
| BAB V Uji COBA DAN EVALUASI..... | | 35 |
| 5.1 | Lingkungan Uji Coba..... | 35 |
| 5.2 | Skenario Uji Coba | 37 |
| 5.2.1 | Uji Fungsionalitas | 37 |
| 5.2.1.1 | Uji Coba pengguna normal mengakses server | 38 |
| 5.2.1.2 | Uji Coba Proses Rekonstruksi Paket Data | 39 |
| 5.2.1.3 | Uji Coba Proses Menghitung N-Gram Paket Data ... | 41 |
| 5.2.1.4 | Uji Coba Proses Membuat Model Data <i>Training</i> | 43 |
| 5.2.1.5 | Uji Coba Sniffing | 44 |
| 5.2.1.6 | Uji Coba Proses Menghitung Jarak Mahalanobis | 45 |
| 5.2.1.7 | Uji Coba Proses Deteksi Paket Data Normal dan Paket Data Intrusi..... | 47 |
| 5.2.1.8 | Uji Coba Proses Incremental Learning | 48 |
| 5.2.2 | Uji Coba Performa..... | 50 |
| 5.2.2.1 | Uji Coba Performa Sistem | 51 |
| 5.2.2.2 | Uji Coba Kecepatan Pendeteksian | 59 |
| 5.2.2.3 | Uji Coba Akurasi | 60 |
| BAB VI KESIMPULAN DAN SARAN..... | | 73 |

| | |
|---|----|
| 6.1 Kesimpulan | 73 |
| 6.2 Saran..... | 73 |
| DAFTAR PUSTAKA | 75 |
| LAMPIRAN..... | 77 |
| A. Kode Sumber..... | 77 |
| A.1. Kode Sumber Proses Rekonstruksi Paket Data..... | 77 |
| A.2. Kode Sumber Proses Penggunaan Metodel N-Gram..... | 83 |
| A.3. Kode Sumber Proses Perancangan Model Data Training.... | 84 |
| A.4. Kode Sumber Sniffing..... | 88 |
| A.5. Kode Sumber Proses Penggunaan Metode Mahalanobis Distance | 91 |
| A.6. Kode Sumber Proses Pendeteksian Serangan | 92 |
| A.7. Kode Sumber Proses Incremental Learning | 96 |
| BIODATA PENULIS | 99 |

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 2.1 Contoh penggunaan Jpcap..... | 9 |
| Gambar 2.2 Kode sumber penggunaan Jpcap untuk <i>offline capture</i> | 9 |
| Gambar 2.3 Contoh keluaran <i>offline capture</i> | 10 |
| Gambar 2.4 Arsitektur DARPA 1999 | 16 |
| Gambar 3.1 Diagram Alir kerja sistem secara umum | 21 |
| Gambar 3.2 Topologi jaringan yang akan digunakan | 22 |
| Gambar 3.3 Proses training data set..... | 23 |
| Gambar 3.4 Proses <i>sniffing</i> | 24 |
| Gambar 3.5 Proses Identifikasi Serangan..... | 25 |
| Gambar 3.6 Contoh <i>file</i> konfigurasi..... | 26 |
| Gambar 3.7 Contoh log hasil luaran sistem | 26 |
| Gambar 4.1 <i>Pseudocode</i> untuk Rekonstruksi paket data | 30 |
| Gambar 4.2 <i>Pseudocode</i> untuk menghitung N-Gram paket data | 31 |
| Gambar 4.3 <i>Pseudocode</i> untuk membuat model data <i>training</i> ... | 31 |
| Gambar 4.4 <i>Pseudocode</i> untuk <i>sniffer</i> | 32 |
| Gambar 4.5 <i>Pseudocode</i> penggunaan metode Mahalanobis Distance | 33 |
| Gambar 4.6 <i>Pseudocode</i> untuk Pendeteksian Serangan..... | 33 |
| Gambar 4.7 <i>Pseudocode</i> untuk <i>Incremental Learning</i> | 34 |
| Gambar 5.1 Luaran yang dihasilkan oleh komputer pengkases normal dengan IP:192.168.57.2..... | 38 |
| Gambar 5.2 Luaran yang dihasilkan oleh komputer penyerang dengan IP:192.168.57.3 | 39 |
| Gambar 5.3 Potongan hasil paket data tanpa rekonstruksi | 40 |
| Gambar 5.4 Potongan hasil paket data setelah direkonstruksi | 41 |
| Gambar 5.5 Potongan hasil N-Gram paket data..... | 42 |
| Gambar 5.6 Potongan hasil model data <i>training</i> | 44 |
| Gambar 5.7 Potongan hasil <i>sniffing</i> | 45 |
| Gambar 5.8 Potongan hasil menghitung jarak mahalanobis | 46 |
| Gambar 5.9 Potongan hasil deteksi paket data normal dan paket data berupa intrusi..... | 48 |

| | |
|--|----|
| Gambar 5.10 Potongan hasil data sebelum proses incremental learning..... | 49 |
| Gambar 5.11 Potongan hasil data setelah proses incremental learning..... | 50 |
| Gambar 5.12 HTOP CPU ketika sistem belum berjalan | 51 |
| Gambar 5.13 HTOP CPU ketika <i>training</i> data set berjalan..... | 51 |
| Gambar 5.14 HTOP CPU ketika identifikasi berjalan..... | 52 |
| Gambar 5.15 Grafik persentase utilisasi CPU | 52 |
| Gambar 5.16 HTOP RAM ketika sistem belum berjalan..... | 53 |
| Gambar 5.17 HTOP RAM ketika <i>training</i> data set berjalan | 53 |
| Gambar 5.18 HTOP RAM ketika identifikasi berjalan..... | 53 |
| Gambar 5.19 Grafik persentase utilisasi RAM | 54 |
| Gambar 5.20 Tampilan halaman web yang akan diakses..... | 55 |
| Gambar 5.21 Luaran ApacheBench untuk skenario 1 | 56 |
| Gambar 5.22 Luaran ApacheBecnh untuk skenario 2 | 57 |
| Gambar 5.23 Luaran ApacheBench untuk skenario 3 | 58 |
| Gambar 5.24 Grafik waktu akses web..... | 59 |
| Gambar 5.25 Grafik durasi waktu pendeteksian serangan | 60 |
| Gambar 5.26 Model Confussion Matrix untuk pengujian | 62 |

DAFTAR TABEL

| | |
|---|----|
| Tabel 2.1 Format data kasar didalam Mahalanobis <i>Distance</i> | 12 |
| Tabel 4.1 Data set file Paket Data | 29 |
| Tabel 4.2 Daftar bagian paket yang dibutuhkan program | 30 |
| Tabel 5.1 Prosedur pengguna normal mengakses server..... | 38 |
| Tabel 5.2 Prosedur rekonstruksi paket data | 39 |
| Tabel 5.3 Prosedur menghitung N-Gram paket data..... | 41 |
| Tabel 5.4 Prosedur membuat model data <i>training</i> | 43 |
| Tabel 5.5 Prosedur <i>sniffing</i> | 44 |
| Tabel 5.6 Prosedur menghitung jarak mahalanobis | 46 |
| Tabel 5.7 Prosedur deteksi paket data normal dan paket data berupa intrusi | 47 |
| Tabel 5.8 Prosedur proses incremental learning..... | 49 |
| Tabel 5.9 Metode akses komputer penyerang..... | 60 |
| Tabel 5.10 Data uji..... | 61 |
| Tabel 5.11 Hasil Uji Data <i>Training</i> minimum jarak paket data intrusi | 64 |
| Tabel 5.12 Hasil Uji Data <i>Training</i> maksimum jarak paket data normal | 64 |
| Tabel 5.13 Threshold untuk masing-masing port | 65 |
| Tabel 5.14 Hasil Uji Data <i>Testing</i> minggu ke-5 tanpa proses <i>incremental learning</i> | 65 |
| Tabel 5.15 <i>Confussion matrix</i> uji coba 1a | 65 |
| Tabel 5.16 Hasil penilaian percobaan dengan ukuran window 10000..... | 66 |
| Tabel 5.17 hasil penilaian percobaan dengan ukuran window 20000..... | 66 |
| Tabel 5.18 Hasil Uji Data <i>Testing</i> minggu ke-5 dengan proses <i>incremental laeraning</i> | 66 |
| Tabel 5.19 <i>Confussion matrix</i> uji coba 1b..... | 67 |
| Tabel 5.20 Hasil penilaian percobaan dengan ukuran window 10000..... | 67 |
| Tabel 5.21 hasil penilaian percobaan dengan ukuran window 20000..... | 67 |

| | |
|--|----|
| Tabel 5.22 Skenario serangan..... | 68 |
| Tabel 5.23 Hasil Uji Data <i>Testing</i> secara <i>real-time</i> | 68 |
| Tabel 5.24 <i>Confussion matrix</i> uji coba 2a..... | 68 |
| Tabel 5.25 Hasil penilaian percobaan FTP <i>brute force</i> | 69 |
| Tabel 5.26 Hasil penilaian percobaan Telnet <i>brute force</i> | 69 |
| Tabel 5.27 Skenario serangan..... | 69 |
| Tabel 5.28 Hasil Uji Data <i>Testing</i> secara <i>real-time</i> | 70 |
| Tabel 5.29 <i>Confussion matrix</i> uji coba 2b..... | 70 |
| Tabel 5.30 Hasil penilaian percobaan FTP <i>brute force</i> | 70 |
| Tabel 5.31 Hasil penilaian percobaan Telnet <i>brute force</i> | 71 |

DAFTAR PERSAMAAN

| | |
|---------------------|----|
| Persamaan 2.1 | 12 |
| Persamaan 2.2 | 13 |
| Persamaan 2.3 | 13 |
| Persamaan 2.4 | 14 |
| Persamaan 2.5 | 14 |
| Persamaan 5.1 | 61 |
| Persamaan 5.2 | 63 |
| Persamaan 5.3 | 63 |
| Persamaan 5.4 | 63 |
| Persamaan 5.5 | 63 |
| Persamaan 5.6 | 63 |
| Persamaan 5.7 | 64 |

[Halaman ini sengaja dikosongkan]

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 serangan 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.

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 tersebut 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.

Ada beberapa teknik yang bisa digunakan untuk memodelkan sebuah paket data. Agar model tersebut terlihat sederhana dan cepat untuk dihitung khususnya menghitung distribusi karakter pada suatu paket data, metode *n-gram* yang

paling tepat. Metode *n-gram* merupakan metode yang efisien dan efektif dalam membuat model dari suatu paket data.

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. Metode mahalanobis *distance* berguna untuk membedakan paket-paket data berdasarkan anomali yang terjadi.

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 yang dapat menangkap paket data dari *network interface* suatu komputer?
3. Bagaimana cara mengklasifikasikan paket data menjadi dua kelompok, yaitu paket data normal dan paket data yang berupa intrusi dengan menggunakan metode Mahalanobis distance?

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 serangan pada jaringan komputer berbasis 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 penggunaan 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.

[Halaman ini sengaja dikosongkan]

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 menyiapkan tindakan untuk menghadapi sebuah serangan. Hal ini dijalankan dengan cara mengumpulkan informasi dari berbagai sumber, baik dari suatu sistem maupun jaringan, lalu menganalisisnya 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 mirip 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 berbasis 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 [2] 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*.


```

Internet Protocol, Src: 125.216.245.23 (125.216.245.23), Dst: 202.112.26.254 (202.112.26.254)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... 00.. = ECN-Capable Transport (ECT): 0
      .... 00.. = ECN-CE: 0
  Total Length: 80
  Identification: 0x3217 (12823)
  Flags: 0x00
    0... = Reserved bit: Not set
    .0.. = Don't Fragment: Not set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 128
  Protocol: IPv6 (0x29)
  Header checksum: 0xb00f [correct]
    [Good: True]
    [Bad: False]
  Source: 125.216.245.23 (125.216.245.23)
  Destination: 202.112.26.254 (202.112.26.254)
Internet Protocol Version 6
  Version: 6
  Traffic class: 0x00
  Flowlabel: 0x000000
  Payload length: 20
  Next header: TCP (0x06)
  Hop limit: 64
  Source address: 2001:da8:8000:3:0:5efe:7dd8:f517

```

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

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 [3] 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 :

| | |
|------------------|--------------|
| <i>uni-gram</i> | : T, E, X, T |
| <i>bi-gram</i> | : TE, EX, XT |
| <i>tri-gram</i> | : TEX, EXT |
| <i>quad-gram</i> | : TEXT, EXT_ |

dan seterusnya.

Sedangkan pada pembangkit kata, metode *n-gram* ini digunakan untuk mengambil potongan kata sejumlah *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 :

| | |
|-----------------|--|
| <i>uni-gram</i> | : saya, dapat , melihat, cahaya, itu |
| <i>bi-gram</i> | : saya dapat, dapat melihat, melihat cahaya, cahaya itu |
| <i>tri-gram</i> | : saya dapat melihat, dapat melihat cahaya, melihat cahaya itu_ |

dan seterusnya.

Salah satu keunggulan menggunakan *n-gram* dan bukan suatu kata utuh secara keseluruhan adalah 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 menghitung 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.

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} (|x_i - \bar{y}_i| / (\bar{\sigma}_i + \alpha)) \quad (2.1)$$

dimana,

d = jarak mahalanobis

x_i = variable ke-i dari *payload* baru

\bar{y}_i = rata-rata variable ke-i dari model data *training*

$\bar{\sigma}_i$ = 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 | \bar{X}_1 | \bar{X}_2 | ... | \bar{X}_i | ... | \bar{X}_{p-1} | \bar{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 menghitung 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:

$$\bar{x} = \frac{\bar{x} \times N + x_{N+1}}{N + 1} = \bar{x} + \frac{x_{N+1} - \bar{x}}{N + 1} \quad (2.4)$$

dimana,

\bar{x} = rata-rata baru

x_{N+1} = nilai dari variabel yang baru

N = jumlah sampel sebelumnya

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

$$S_i = \sqrt{\frac{(n + 1) \times (\sum_{i=1}^n x_i^2 + x_{n+1}^2) - (\sum_{i=1}^n x_i + x_{n+1})^2}{(n + 1)n}} \quad (2.5)$$

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 [4] 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 serangan. Data set ini merupakan salah satu yang *dump file* dengan jenis serangan terlengkap sehingga data set ini sering digunakan untuk penelitian khususnya penelitian yang berhubungan dengan IDS (*Intrusion Detection System*). Beberapa publikasi ilmiah yang menggunakan data set ini antara lain:

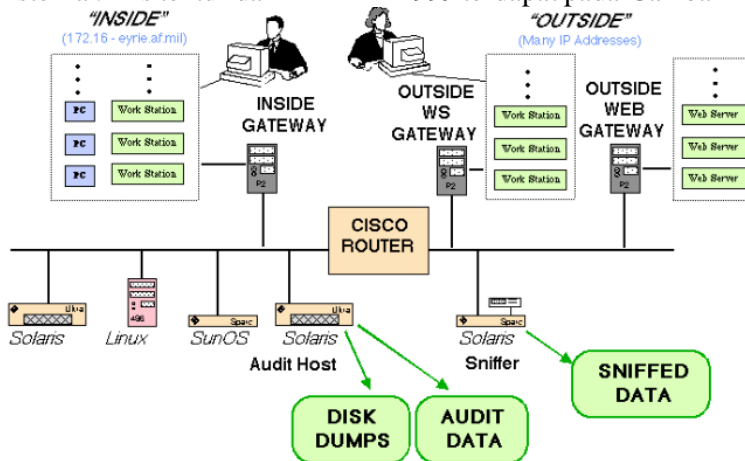
- a. Richard P. Lippmann, Robert K. Cunningham, David J. Fried, Issac Graf, Kris R. Kendala, Seth E. Webster, Marc A. Zissman, "Results of the DARPA 1998 Offline Intrusion Detection Evaluation", dipresentasikan di *RAID 1999 Conference*, September 7-9, 1999, West Lafayette, Indiana.
- b. Richard P. Lippmann and Robert K. Cunningham, "Using Key-String Selection and Neural Networks to Reduce False Alarms and Detect New Attacks with Sniffer-Based Intrusion Detection System", 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 beberapa 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 yang 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 penggunaanya. 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 misal 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 dalam sistem terlebih dahulu sebagai pengguna biasa.

[Halaman ini sengaja dikosongkan]

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 serangan 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* selanjutnya 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.

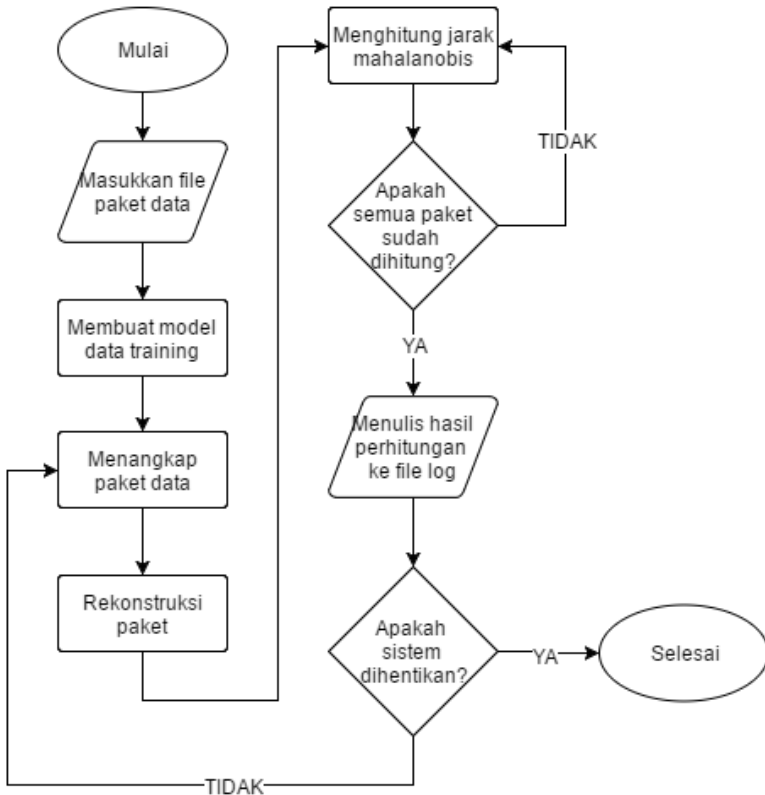
3.2.1 Alur Kerja Sistem Secara Umum

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.

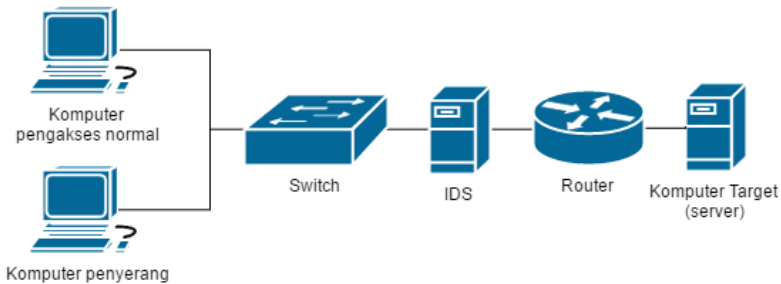
Proses diatas diulangi sampai aplikasi dihentikan oleh pengguna dan menulis hasil keputusan terhadap proses perbandingan ke sebuah *file log*.



Gambar 3.1 Diagram Alir kerja sistem secara umum

3.2.2 Perancangan Arsitektur Jaringan

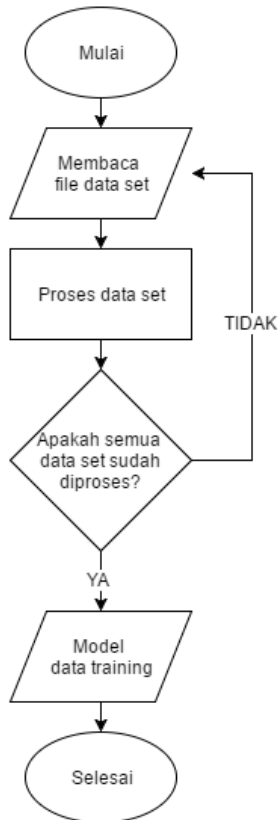
Sistem pendeteksi ini bekerja pada suatu arsitektur jaringan dengan multi subnet yang diilustrasikan pada Gambar 3.2. IDS akan menangkap paket data yang datang maupun keluar dari internet. IDS bekerja sesuai dengan konfigurasi yang dilakukan oleh pengguna. Konfigurasi tersebut disimpan dalam sebuah *file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

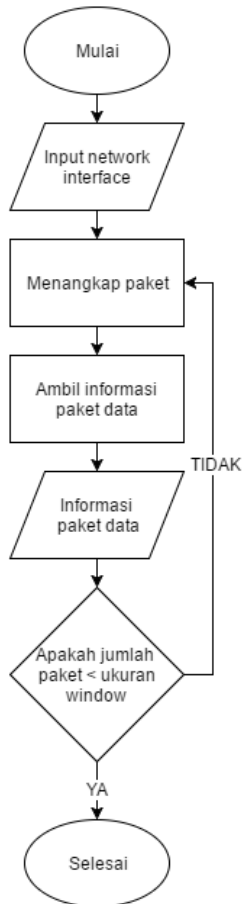


Gambar 3.3 Proses training data set

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

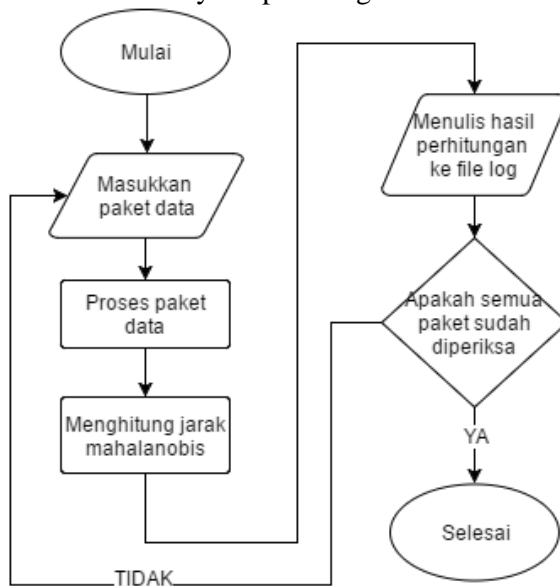
memilih network interface yang akan dimonitor. Selajutnya sistem akan membaca konfigurasi yang terdapat pada file *conf*. Pada file konfigurasi terdapat jumlah paket data yang akan ditangkap oleh sniffer. Jika sniffer sudah menangkap paket data sejumlah yang ditentukan pada file konfigurasi, sniffer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



Gambar 3.4 Proses *sniffing*

3.2.5 Perancangan Proses Identifikasi Serangan

Pada bagian ini akan dijelaskan cara program melakukan deteksi serangan. 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.



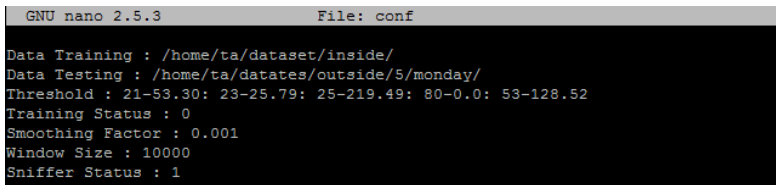
Gambar 3.5 Proses Identifikasi Serangan

Pada Gambar 3.5 menjabarkan mengenai alur kerja proses deteksi serangan. Pertama akan dijalankan pemeriksaan apakah Port tujuan paket sama dengan Port tujuan data *training*. Lalu mengambil rata-rata dan standar deviasi yang ada pada model sesuai dengan port paket. Proses selanjutnya 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 serangan. Lalu program akan menulis informasi paket ke sebuah file log.

3.2.6 Rancangan Antarmuka

Sistem pendeteksi serangan 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/dataset/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 dilakukan 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 sebagai 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) i3-2120 2.40GHz 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 serangan. 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 terdapat 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 Penangkapan 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> datasetUdp
3 Terima paket menggunakan fungsi getPacket()
4 Periksa tipe protokol paket
5 Jika paket termasuk TCP dan Port tujuan kurang
  dari 1024
6     Cocokkan IP asal, Port asal, IP tujuan dan
      Port tujuan
7     Tambahkan paket ke datasetTcp
8 Jika paket termasuk UDP dan Port tujuan kurang
  dari 1024
9     Cocokkan 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 penggunaan 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 menghitung *N-Gram* dapat dilihat pada Gambar 4.2.

```

1 Terima konten paket menggunakan fungsi Ngram()
2 Jika konten paket tidak kosong
3 Deklarasi double[] n = new double[256];
4 Baca konten paket
5 Konversi konten paket data menjadi unsigned
  integer

```

| | |
|---|--|
| 6 | 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 serangan. 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 model data *training* dapat dilihat pada Gambar 4.3.

| | |
|----|--|
| 1 | Deklarasi ArrayList<Double[]> dataTraining |
| 2 | Deklarasi ArrayList<DataPacket> datasetTcp |
| 3 | Deklarasi ArrayList<DataPacket> datasetUdp |
| 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 menggunakan protokol TCP atau UDP. Paket yang menggunakan selain protokol TCP atau UDP tidak ditangkap atau dibiarkan lewat. *Pseudocode* untuk *sniffer* dapat 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> datasetUdp
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 Serangan

Implementasi pendeteksian serangan 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 serangan 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 serangan
5   Tulis informasi paket ke file log

```

Gambar 4.6 Pseudocode untuk Pendeteksian Serangan

4.2.8 Implementasi Proses *Incremental Learning*

Implementasi proses incremental learning digunakan untuk memperbaharui model data *training*. Nilai yang diperbaharui dari model, yaitu 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

BAB V

UJI COBA DAN EVALUASI

Pada bab ini akan membahas uji coba dan evaluasi dari sistem yang dibuat. Sistem akan diuji coba fungsionalitas dan performa dengan menjalankan skenario yang sudah ditentukan. Uji coba dilakukan untuk mengetahui hasil dari sistem ini sehingga menjawab rumusan masalah pada tugas akhir ini.

5.1 Lingkungan Uji Coba

Uji coba aplikasi ini dilakukan dengan menggunakan satu buah komputer target, satu buah komputer difungsikan sebagai *router* dan juga sebagai tempat berjalannya sistem yang dibuat, satu buah komputer pengakses normal serta satu buah komputer penyerang.

1. Router komputer

- Spesifikasi perangkat keras
 - Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
 - 8GB RAM
- Spesifikasi perangkat lunak
 - Sistem Operasi Linux Ubuntu Desktop 16.04 LTS 64-bit
 - Oracle Java Development Kit (JDK) 1.8 (64-bit)
 - Jpcap 0.7
- Konfigurasi jaringan
 - IP address: 172.16.2.1
 - Netmask: 255.255.255.0
 - Network: 172.16.2.0

 - IP address: 192.168.57.1
 - Netmask: 255.255.255.0
 - Network: 192.168.57.0

2. Komputer target

- Spesifikasi perangkat keras
 - Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
 - 1GB RAM
- Spesifikasi perangkat lunak
 - Sistem Operasi Linux Ubuntu Server 14.04 LTS 64-bit
 - Apache web server
 - FTP server
 - MySQL server
 - SMTP server
 - DNS server
- Konfigurasi jaringan
 - IP address: 172.16.2.2
 - Netmask: 255.255.255.0
 - Gateway: 172.16.2.1

3. Komputer pengakses normal

- Spesifikasi perangkat keras
 - Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
 - 1GB RAM
- Spesifikasi perangkat lunak
 - Sistem Operasi Linux Ubuntu Server 14.04 LTS 64-bit
 - ApacheBench
- Konfigurasi jaringan
 - IP address: 192.168.57.2
 - Netmask: 255.255.255.0
 - Gateway: 192.168.57.1

4. Komputer penyerang

- Spesifikasi perangkat keras
 - Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
 - 1GB RAM
- Spesifikasi perangkat lunak

- Sistem Operasi Linux Ubuntu Server 14.04 LTS 64-bit
- WPScan
- Patator
- SQLMap
- Konfigurasi jaringan
 - IP address: 192.168.57.3
 - Netmask: 255.255.255.0
 - Gateway: 192.168.57.1

5.2 Skenario Uji Coba

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Beberapa skenario akan dilakukan guna memastikan aplikasi ini sudah siap berjalan sesuai dengan fungsinya atau tidak.

Pengujian akan dibagi menjadi dua bagian. Bagian yang pertama adalah uji coba fungsionalitas, pada bagian ini aplikasi akan diuji coba per kebutuhan fungsionalitas sistem sesuai dengan rancangan implementasi yang telah disampaikan pada Bab IV. Tujuan dari uji coba fungsionalitas adalah memastikan bahwa kebutuhan dari aplikasi ini sudah terpenuhi.

Bagian kedua adalah uji coba performa. Pada bagian ini aplikasi akan diukur seberapa besar sistem yang dijalankan mempengaruhi performa dari komputer tempat sistem ini terpasang dan seberapa cepat dalam mendeteksi serangan serta berapa akurasi yang dihasilkan dalam mendeteksi paket data normal maupun paket data yang berupa intrusi.

5.2.1 Uji Fungsionalitas

Uji coba fungsionalitas adalah pemeriksaan apakah rancangan dan implementasi yang sudah dijelaskan pada Bab III dan Bab IV sudah berjalan sesuai rencana. Pembagian subbab ini berdasarkan rancangan dan implementasi tersebut.

5.2.1.1 Uji Coba pengguna normal mengakses server

Uji coba ini dilakukan dengan cara seluruh komputer mengakses normal dan komputer penyerang melakukan ping test dengan tujuan server target. Kegiatan ini dilakukan hingga pada masing-masing komputer mengakses normal dan komputer penyerang mendapatkan 4 balasan ping oleh server tujuan. Pada Tabel 5.1 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.1 Prosedur pengguna normal mengakses server

| | |
|------------------------|---|
| ID | UJ-01 |
| Nama | Uji coba pengguna normal mengakses server |
| Tujuan Uji Coba | Menguji apakah komputer pengakses normal dan komputer penyerang dapat saling berkomunikasi dengan server target |
| Kondisi Awal | Konfigurasi jaringan pada komputer pengakses normal, komputer penyerang dan server target sudah dilakukan |
| Skenario | Komputer pengakses normal dan komputer penyerang melakukan ping ke IP server target |
| Masukan | IP server target |
| Keluaran | Balasan dari server target |
| Hasil Uji Coba | Berhasil |

```
ta@singa:~$ ping -c 4 172.16.2.2
PING 172.16.2.2 (172.16.2.2) 56(84) bytes of data:
64 bytes from 172.16.2.2: icmp_seq=1 ttl=63 time=0.381 ms
64 bytes from 172.16.2.2: icmp_seq=2 ttl=63 time=0.270 ms
64 bytes from 172.16.2.2: icmp_seq=3 ttl=63 time=0.282 ms
64 bytes from 172.16.2.2: icmp_seq=4 ttl=63 time=0.299 ms

--- 172.16.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.270/0.308/0.381/0.043 ms
ta@singa:~$ _
```

Gambar 5.1 Luaran yang dihasilkan oleh komputer pengakses normal dengan IP:192.168.57.2

Pada Gambar 5.1 menunjukkan luaran yang muncul pada *terminal* komputer pengakses normal sedangkan Gambar 5.2 menunjukkan luaran yang muncul pada *terminal* komputer penyerang.

```
ta@ga.jah:~$ ping -c 4 172.16.2.2
PING 172.16.2.2 (172.16.2.2) 56(84) bytes of data.
64 bytes from 172.16.2.2: icmp_seq=1 ttl=63 time=0.276 ms
64 bytes from 172.16.2.2: icmp_seq=2 ttl=63 time=0.416 ms
64 bytes from 172.16.2.2: icmp_seq=3 ttl=63 time=0.406 ms
64 bytes from 172.16.2.2: icmp_seq=4 ttl=63 time=0.268 ms

--- 172.16.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.268/0.341/0.416/0.072 ms
ta@ga.jah:~$ _
```

Gambar 5.2 Luaran yang dihasilkan oleh komputer penyerang dengan IP:192.168.57.3

5.2.1.2 Uji Coba Proses Rekonstruksi Paket Data

Uji coba rekonstruksi paket data dilakukan dengan cara membaca sebuah file paket data lalu menampilkan hasil tangkapan paket data yang belum direkonstruksi dan paket data yang sudah direkonstruksi pada *console*. Pada Tabel 5.2 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.2 Prosedur rekonstruksi paket data

| | |
|------------------------|--|
| ID | UJ-02 |
| Nama | Uji coba rekonstruksi paket data |
| Tujuan Uji Coba | Menguji apakah aplikasi dapat mengelompokkan paket data yang memiliki protokol, IP asal, Port asal, IP tujuan, serta Port tujuan yang sama menjadi sebuah paket data yang baru |
| Kondisi Awal | Aplikasi berjalan, rekonstruksi paket data dimulai |
| Skenario | Menangkap paket data dengan library Jpcap lalu mencocokkan setiap paket data yang memiliki protokol, IP asal, Port asal, IP tujuan serta Port tujuan yang sama |
| Masukan | File paket data |

| | |
|-----------------------|--------------------------------------|
| Keluaran | Paket data yang telah direkonstruksi |
| Hasil Uji Coba | Berhasil |

Pada Gambar 5.3 ditunjukkan potongan hasil paket data yang belum diolah, dimana setiap paket data memiliki satu paket header dan satu konten paket. Paket header ditandai dengan warna merah dan konten paket ditandai dengan warna hijau pada gambar. Sedangkan pada Gambar 5.4 ditunjukkan potongan hasil paket data yang sudah direkonstruksi. Supaya hasil rekonstruksi paket data dapat ditampilkan untuk memastikan paket data tersebut sudah direkonstruksi atau tidak maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

```
Paket data ke-1 -> 1254030344:418804 /192.168.1.2->/74.125.67.100 protocol(6) priority(0) hop(128)
offset(0) ident(11339) TCP 1449 > 80 seq(1762339492) win(64604) ack 3683340383 P
GET /generate 204 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko)
Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/search?hl=en&source=hp&q=wireshark&btnG=Google+Search&meta=&aq=f&oq=
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

Paket data ke-10 -> 1254030345:88934 /192.168.1.2->/74.125.19.103 protocol(6) priority(0) hop(128)
offset(0) ident(11352) TCP 1447 > 80 seq(215618439) win(65535) ack 1904946141 P
GET /csi?v=3&s=web&action=&srt=1164&tran=undefined&=17259,21589,21766,21819,22023&ei=A_y-
StKoEY6Qsg0suKF0&rt=sprt.40,xjs.161,ol.814 HTTP/1.1
Host: www.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko)
Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/search?hl=en&source=hp&q=wireshark&btnG=Google+Search&meta=&aq=f&oq=
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

Paket data ke-20 -> 1254030339:430292 /192.168.1.2->/74.125.67.100 protocol(6) priority(0) hop(128)
offset(0) ident(11274) TCP 1449 > 80 seq(1762337759) win(65535) ack 3683339452 P
GET /complete/search?hl=en&q=wire&cp=4 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko)
Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Gambar 5.3 Potongan hasil paket data tanpa rekonstruksi


```

Paket data ke-1 -> TCP 192.168.1.2:1449-74.125.67.100:80
GET /complete/search?hl=en&q=wireshark&cp=4 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

GET /complete/search?hl=en&q=wireshark&cp=9 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

GET /generate 204 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/search?hl=en&source=hp&q=wireshark&btnG=Google+Search&meta=&aq=f&oq=
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

```

Gambar 5.4 Potongan hasil paket data setelah direkonstruksi

5.2.1.3 Uji Coba Proses Menghitung N-Gram Paket Data

Uji coba menghitung n-gram paket data dilakukan dengan cara memanggil fungsi Ngram pada kelas Ngram dengan parameter konten paket data dalam bentuk *array of byte*. Pada Tabel 5.3 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.3 Prosedur menghitung N-Gram paket data

| | |
|------------------------|---|
| ID | UJ-03 |
| Nama | Uji coba menghitung n-gram paket data |
| Tujuan Uji Coba | Menguji apakah jumlah distribusi byte setiap karakter dari konten paket data sesuai dengan panjang paket data |
| Kondisi Awal | Aplikasi berjalan, paket data telah direkonstruksi |
| Skenario | Aplikasi menghitung jumlah setiap karakter yang ada pada konten paket data |
| Masukan | Konten paket data dalam bentuk <i>array of byte</i> |

| | |
|-----------------------|---|
| Keluaran | Frekuensi setiap karakter dalam bentuk <i>array of double</i> |
| Hasil Uji Coba | Berhasil |

Pada Gambar 5.5 ditunjukkan potongan hasil perhitungan N-Gram paket data. Uji coba ini melibatkan model perhitungan menggunakan 1-gram. Setiap paket data baru memiliki model 1-gram sendiri. Paket header ditandai dengan warna merah dan n-gram dari konten paket ditandai dengan warna hijau pada gambar. Model 1-gram antara paket data mungkin saja memiliki kemiripan atau bahkan sama, itu dikarenakan konten dari paket data tersebut terdiri dari kumpulan karakter yang sama. Supaya hasil perhitungan N-Gram paket data dapat ditampilkan untuk memastikan panjang konten paket data dan dengan jumlah karakter hasil n-gram sama atau tidak maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

```
TCP 192.168.1.2:1447-74.125.19.103:80
Panjang konten paket data : 2027
N-Gram :
[0.0, 22.0, 22.0, 53.0, 19.0, 4.0, 4.0, 6.0, 3.0, 21.0, 35.0, 40.0, 25.0, 37.0, 45.0, 43.0, 31.0, 25.0,
40.0, 22.0, 22.0, 42.0, 27.0, 28.0, 22.0, 53.0, 3.0, 48.0, 6.0, 31.0, 23.0, 17.0, 8.0, 15.0, 22.0, 25.0,
15.0, 14.0, 12.0, 15.0, 13.0, 13.0, 18.0, 12.0, 11.0, 31.0, 19.0, 22.0, 14.0, 16.0, 6.0, 4.0, 9.0, 20.0,
50.0, 17.0, 53.0, 16.0, 94.0, 31.0, 31.0, 25.0, 47.0, 17.0, 22.0, 41.0, 17.0, 50.0, 59.0, 35.0, 27.0,
38.0, 33.0, 55.0, 14.0, 16.0, 37.0, 22.0, 10.0, 17.0]
Jumlah karakter : 2027.0

TCP 192.168.1.2:1352-74.125.67.83:443
Panjang konten paket data : 1797
N-Gram :
[11.0, 13.0, 9.0, 10.0, 6.0, 9.0, 7.0, 3.0, 4.0, 6.0, 11.0, 6.0, 4.0, 6.0, 9.0, 10.0, 4.0, 9.0, 3.0,
5.0, 8.0, 10.0, 4.0, 8.0, 7.0, 4.0, 9.0, 5.0, 7.0, 8.0, 7.0, 5.0, 6.0, 13.0, 6.0, 11.0, 12.0, 8.0, 7.0,
13.0, 9.0, 9.0, 7.0, 5.0, 12.0, 11.0, 12.0, 5.0, 8.0, 11.0, 8.0, 8.0, 8.0, 4.0, 8.0, 9.0, 10.0,
6.0, 10.0, 7.0, 9.0, 1.0, 2.0, 8.0, 4.0, 5.0, 5.0, 6.0, 9.0, 6.0, 8.0, 12.0, 8.0, 11.0, 10.0, 10.0,
15.0, 8.0, 5.0, 7.0, 7.0, 9.0, 5.0, 7.0, 6.0, 3.0, 8.0, 12.0, 6.0, 11.0, 11.0, 4.0, 10.0, 9.0, 6.0,
4.0, 7.0, 2.0, 9.0, 6.0, 10.0, 2.0, 8.0, 3.0, 6.0, 6.0, 8.0, 7.0, 4.0, 7.0, 9.0, 6.0, 6.0, 3.0, 6.0,
8.0, 8.0, 8.0, 6.0, 5.0, 9.0, 5.0, 8.0, 1.0, 5.0, 8.0, 5.0, 4.0, 8.0, 5.0, 7.0, 7.0, 5.0, 7.0, 10.0,
13.0, 6.0, 8.0, 5.0, 5.0, 5.0, 3.0, 8.0, 6.0, 5.0, 9.0, 9.0, 5.0, 5.0, 4.0, 3.0, 2.0, 7.0, 6.0, 5.0,
11.0, 7.0, 6.0, 8.0, 8.0, 7.0, 7.0, 6.0, 2.0, 4.0, 5.0, 5.0, 11.0, 3.0, 10.0, 7.0, 7.0, 6.0, 4.0, 7.0,
10.0, 6.0, 5.0, 7.0, 4.0, 7.0, 10.0, 7.0, 2.0, 9.0, 10.0, 6.0, 7.0, 7.0, 6.0, 6.0, 14.0, 2.0, 5.0, 10.0,
7.0, 8.0, 8.0, 4.0, 8.0, 9.0, 7.0, 4.0, 10.0, 10.0, 10.0, 11.0, 6.0, 8.0, 7.0, 6.0, 8.0, 6.0, 10.0, 6.0,
7.0, 8.0, 7.0, 2.0, 9.0, 8.0, 9.0, 7.0, 8.0, 6.0, 5.0, 3.0, 6.0, 7.0, 12.0, 9.0, 6.0, 7.0, 6.0, 7.0,
7.0, 4.0, 6.0, 3.0, 7.0, 7.0, 3.0, 5.0, 4.0, 10.0, 7.0, 10.0, 13.0, 9.0, 3.0, 6.0, 6.0, 2.0, 7.0]
Jumlah karakter : 1797.0

TCP 192.168.1.2:1449-74.125.67.100:80
Panjang konten paket data : 2644
N-Gram :
[0.0, 33.0, 33.0, 78.0, 10.0, 6.0, 6.0, 9.0, 1.0, 15.0, 51.0, 51.0, 32.0, 49.0, 60.0, 46.0, 42.0, 35.0,
57.0, 27.0, 30.0, 51.0, 31.0, 42.0, 27.0, 52.0, 3.0, 66.0, 9.0, 48.0, 39.0, 24.0, 6.0, 14.0, 33.0, 36.0,
21.0, 18.0, 21.0, 21.0, 21.0, 18.0, 27.0, 15.0, 18.0, 46.0, 27.0, 36.0, 21.0, 18.0, 9.0, 3.0, 12.0,
25.0, 54.0, 22.0, 75.0, 21.0, 128.0, 43.0, 41.0, 28.0, 60.0, 30.0, 32.0, 51.0, 18.0, 62.0, 75.0, 44.0,
29.0, 41.0, 39.0, 65.0, 13.0, 27.0, 48.0, 24.0, 18.0, 27.0]
Jumlah karakter : 2644.0
```

Gambar 5.5 Potongan hasil N-Gram paket data

5.2.1.4 Uji Coba Proses Membuat Model Data *Training*

Uji coba membuat model data *training* dilakukan dengan cara membuat sebuah *object* Data *Training* baru dengan parameter protokol paket data (TCP atau UDP), array list Data Packet TCP, array list Data Packet UDP, array list Data *Training* TCP, array list Data *Training* UDP, serta port tujuan paket data. Pada Tabel 5.4 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.4 Prosedur membuat model data *training*

| | |
|------------------------|---|
| ID | UJ-04 |
| Nama | Uji coba membuat model data training |
| Tujuan Uji Coba | Menguji apakah sistem dapat membuat sebuah model data training sesuai dengan port tujuan dari paket data. |
| Kondisi Awal | Aplikasi berjalan, n-gram dari paket data |
| Skenario | Aplikasi menghitung rata-rata setiap variable, standar deviasi setiap variable, jumlah kuadrat setiap variabel, dan jumlah paket data |
| Masukan | Port tujuan paket data dan n-gram dari paket data |
| Keluaran | Model data training sesuai dengan port tujuan, dimana port tujuan yang kurang dari 1024 |
| Hasil Uji Coba | Berhasil |

Pada Gambar 5.6 ditunjukkan potongan hasil pembuatan model data *training*. Model data training terdiri dari, port protokol, jumlah *connction*, jumlah setiap variable yang ditandai dengan warna merah, rata-rata variable yang ditandai dengan warna biru, standar deviasi variable yang ditandai dengan warna hijau, dan jumlah kuadrat dari variable ditandai dengan warna oranye. *Supaya* hasil pembuatan model data *training* paket data dapat ditampilkan untuk memastikan rata-rata variabel dan standar deviasi sudah tersimpan atau tidak maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

| |
|---|
| Port tujuan : 80 Total paket data : 4 |
| Jumlah setiap variabel: [88.0, 88.0, 210.0, 39.0, 16.0, 16.0, 24.0, 5.0, 51.0, 137.0, 143.0, 90.0, 136.0, 167.0, 140.0, 117.0, 95.0, 155.0, 78.0, 84.0, 146.0, 89.0, 112.0, 77.0, 160.0, 9.0, 186.0, 26.0, 128.0, 104.0, 68.0, 21.0, 45.0, 88.0, 98.0, 60.0, 51.0, 54.0, 59.0, 57.0, 51.0, 72.0, 45.0, 47.0, 125.0, 74.0, 98.0, 56.0, 54.0, 27.0, 13.0, 33.0, 69.0, 158.0, 61.0, 202.0, 60.0, 348.0, 119.0, 114.0, 82.0, 169.0, 78.0, 85.0, 144.0, 56.0, 172.0, 209.0, 123.0, 86.0, 120.0, 110.0, 183.0, 40.0, 73.0, 134.0, 71.0, 47.0, 72.0] |
| Rata-rata setiap variabel: [22.0, 22.0, 52.5, 9.75, 4.0, 4.0, 6.0, 1.25, 12.75, 34.25, 35.75, 22.5, 34.0, 41.75, 35.0, 29.25, 23.75, 38.75, 19.5, 21.0, 36.5, 22.25, 28.0, 19.25, 40.0, 2.25, 46.5, 6.5, 32.0, 26.0, 17.0, 5.25, 11.25, 22.0, 24.75, 15.0, 12.75, 13.5, 14.75, 14.25, 12.75, 18.0, 11.25, 11.75, 31.25, 18.5, 24.5, 14.0, 13.5, 6.75, 3.25, 8.25, 17.25, 39.5, 15.25, 50.5, 15.0, 87.0, 29.75, 28.5, 20.5, 42.25, 19.5, 21.25, 36.0, 14.0, 43.0, 52.25, 30.75, 21.5, 30.0, 27.5, 45.75, 10.0, 18.25, 33.5, 17.75, 11.75, 18.0] |
| Standar deviasi setiap variabel: [8.98, 8.98, 20.82, 6.65, 1.63, 1.63, 2.44, 1.25, 6.84, 13.88, 13.76, 8.73, 12.83, 15.67, 11.91, 10.68, 9.06, 15.12, 6.75, 7.78, 13.57, 8.77, 11.43, 7.18, 14.89, 0.95, 15.60, 1.73, 12.67, 9.62, 5.35, 2.21, 3.77, 8.98, 8.99, 4.54, 4.57, 5.80, 4.92, 4.99, 4.11, 7.34, 2.98, 4.92, 11.47, 6.95, 8.38, 5.71, 4.43, 1.5, 0.95, 3.30, 7.18, 15.32, 5.85, 20.88, 4.96, 34.30, 10.68, 10.40, 7.32, 15.84, 7.93, 8.61, 13.88, 4.24, 17.77, 20.15, 12.57, 7.93, 11.22, 11.23, 18.99, 4.24, 6.34, 12.87, 6.75, 4.64, 6.97] |
| Jumlah kuadrat setiap variabel: [2178.0, 2178.0, 12326.0, 513.0, 72.0, 72.0, 162.0, 11.0, 791.0, 5271.0, 5681.0, 2254.0, 5118.0, 7709.0, 5326.0, 3765.0, 2503.0, 6693.0, 1658.0, 1946.0, 5882.0, 2211.0, 3528.0, 1637.0, 7066.0, 23.0, 9380.0, 178.0, 4578.0, 2982.0, 1242.0, 125.0, 549.0, 2178.0, 2693.0, 962.0, 713.0, 830.0, 943.0, 887.0, 701.0, 1458.0, 533.0, 625.0, 4301.0, 1514.0, 2612.0, 882.0, 788.0, 189.0, 45.0, 305.0, 1345.0, 6946.0, 1033.0, 11510.0, 974.0, 33806.0, 3883.0, 3574.0, 1842.0, 7893.0, 1710.0, 2029.0, 5762.0, 838.0, 8344.0, 12139.0, 4257.0, 2038.0, 3978.0, 3404.0, 9455.0, 454.0, 1453.0, 4986.0, 1397.0, 617.0, 1442.0] |

Gambar 5.6 Potongan hasil model data *training*

5.2.1.5 Uji Coba Sniffing

Uji coba sniffing dilakukan dengan cara memilih menu “sniffer testing” pada aplikasi sehingga aplikasi akan menangkap paket data pada lalu lintas jaringan. Pada Tabel 5.5 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.5 Prosedur *sniffing*

| | |
|------------------------|--|
| ID | UJ-05 |
| Nama | Uji coba <i>sniffing</i> |
| Tujuan Uji Coba | Menguji apakah sistem dapat menangkap paket yang ada pada lalu lintas jaringan secara <i>real-time</i> . |
| Kondisi Awal | Aplikasi berjalan, menu “sniffer testing” sudah terpilih |
| Skenario | Komputer penyerang mengirimkan <i>request</i> HTTP ke komputer target |
| Masukan | Paket data yang ada pada lalu lintas jaringan |
| Keluaran | Paket data yang berhasil ditangkap |
| Hasil Uji Coba | Berhasil |

Pada Gambar 5.7 ditunjukkan potongan hasil proses *sniffing*. Informasi dari paket data yang disimpan adalah paket *header* yang ditandai dengan warna merah dan konten paket data yang ditandai dengan warna hijau pada gambar. Supaya hasil *sniffing* dapat ditampilkan untuk memastikan *sniffer* dapat menangkap paket data dari *network interface* atau tidak maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

```
Paket data ke-1 -> 1254030344:418804 /192.168.1.2->/74.125.67.100 protocol(6) priority(0) hop(128)
offset(0) ident(11339) TCP 1449 > 80 seq(1762339492) win(64064) ack 3683340383 P
GET /generate 204 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko)
Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/search?hl=en&source=hp&q=wreshark&btnG=Google+Search&meta=&aq=f&oq=
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

Paket data ke-10 -> 1254030345:88934 /192.168.1.2->/74.125.19.103 protocol(6) priority(0) hop(128)
offset(0) ident(11332) TCP 1447 > 80 seq(213618439) win(65535) ack 1904946141 P
GET /csi?v=3&s=web&action=bsrt=1164&tran=undefined&e=17259,21589,21766,21819,22023&ei=A_y-
StKoEY6Qsg0sukFO&rt=prt.40,xjs.161,ol.814 HTTP/1.1
Host: www.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko)
Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/search?hl=en&source=hp&q=wreshark&btnG=Google+Search&meta=&aq=f&oq=
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

Paket data ke 20 -> 1254030339:430292 /192.168.1.2->/74.125.67.100 protocol(6) priority(0) hop(128)
offset(0) ident(11274) TCP 1449 > 80 seq(1762337759) win(65535) ack 3683339452 P
GET /complete/search?hl=en&q=wire&cp=4 HTTP/1.1
Host: clients1.google.co.in
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko)
Chrome/3.0.195.21 Safari/532.0
Referer: http://www.google.co.in/
Accept: */*
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Gambar 5.7 Potongan hasil *sniffing*

5.2.1.6 Uji Coba Proses Menghitung Jarak Mahalanobis

Uji coba menghitung jarak mahalanobis dilakukan dengan cara memanggil fungsi *distance* yang ada pada kelas Mahalanobis dengan parameter *array of double* ngram paket data baru, *array of double* rata-rata model, *array of double* standar deviasi model dan nilai *smoothing factor*. Pada Tabel 5.6 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.6 Prosedur menghitung jarak mahalanobis

| | |
|-----------------|--|
| ID | UJ-06 |
| Nama | Uji coba menghitung jarak mahalanobis |
| Tujuan Uji Coba | Menguji apakah sistem dapat menghitung jarak mahalanobis antara paket baru dengan model data training menggunakan metode Mahalanobis Distance. |
| Kondisi Awal | Aplikasi berjalan, sudah ada n-gram dari paket data baru, sudah ada model data training |
| Skenario | Aplikasi menerima masukan dan menghitung jarak mahalanobis menggunakan metode Mahalanobis Distance |
| Masukan | n-gram paket data baru, rata-rata model, standar deviasi model, dan <i>smoothing factor</i> |
| Keluaran | Nilai jarak mahalanobis |
| Hasil Uji Coba | Berhasil |

```
+++++
# Start time : 2016-July-02 14:39:55 PM #
+++++
Protokol | Date | Source | Destination | Distance
-----
TCP | 27/09/2009 01:45:38 | 192.168.1.2:1442 | 74.125.67.100:80 | 16.93
UDP | 27/09/2009 01:45:44 | 192.168.1.2:64505 | 192.168.1.1:53 | 37.57
UDP | 27/09/2009 01:45:44 | 192.168.1.2:49837 | 192.168.1.1:53 | 27.21
UDP | 27/09/2009 01:45:44 | 192.168.1.2:64171 | 192.168.1.1:53 | 34.04
UDP | 27/09/2009 01:45:44 | 192.168.1.2:51358 | 192.168.1.1:53 | 29.0
UDP | 27/09/2009 01:45:44 | 192.168.1.2:52142 | 192.168.1.1:53 | 35.51
UDP | 27/09/2009 01:45:44 | 192.168.1.2:64911 | 192.168.1.1:53 | 28.64
TCP | 27/09/2009 01:45:43 | 192.168.1.2:1447 | 74.125.19.103:80 | 27.23
UDP | 27/09/2009 01:45:44 | 192.168.1.2:53787 | 192.168.1.1:53 | 34.74
TCP | 27/09/2009 01:45:39 | 192.168.1.2:4491 | 74.125.67.19:443 | 163.22
TCP | 27/09/2009 01:45:39 | 192.168.1.2:1449 | 74.125.67.100:80 | 86.92
UDP | 27/09/2009 01:45:44 | 192.168.1.2:58935 | 192.168.1.1:53 | 40.26
```

Gambar 5.8 Potongan hasil menghitung jarak mahalanobis

Pada Gambar 5.8 ditunjukkan potongan hasil perhitungan jarak mahalanobis setiap paket data. Supaya hasil perhitungan jarak mahalanobis paket data dapat ditampilkan untuk memastikan sistem dapat menghitung jarak mahalanobis antara paket dengan data training atau tidak maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

5.2.1.7 Uji Coba Proses Deteksi Paket Data Normal dan Paket Data Intrusi

Uji coba deteksi paket data normal dan paket data yang berupa intrusi dilakukan dengan membandingkan jarak mahalanobis paket data baru dengan nilai *threshold* yang sudah ditentukan sebelumnya. Pada Tabel 5.7 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.7 Prosedur deteksi paket data normal dan paket data berupa intrusi

| | |
|------------------------|---|
| ID | UJ-07 |
| Nama | Uji coba deteksi paket data normal dan paket data yang berupa intrusi |
| Tujuan Uji Coba | Menguji apakah sistem dapat mendeteksi paket data normal dan paket data yang berupa intrusi |
| Kondisi Awal | Aplikasi berjalan, membandingkan jarak mahalanobis dengan <i>threshold</i> dimulai |
| Skenario | Aplikasi membandingkan beberapa paket data dengan <i>threshold</i> |
| Masukan | Nilai jarak mahalanobis paket data dan nilai <i>threshold</i> |
| Keluaran | Paket data tersebut berupa intrusi atau paket data normal |
| Hasil Uji Coba | Berhasil |

Pada Gambar 5.9 ditunjukkan potongan hasil deteksi paket data normal maupun paket data yang berupa intrusi. Supaya hasil deteksi paket data dapat ditampilkan untuk memastikan sistem dapat mengklasifikasikan paket data menjadi dua kelompok yaitu, paket data normal dan paket data intrusi maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

```
+++++
# Start time : 2016-July-02 14:39:55 PM #
+++++
Protokol | Date | Source | Destination | Keterangan
-----
TCP | 27/09/2009 01:45:38 | 192.168.1.2:1442 | 74.125.67.100:80 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:64505 | 192.168.1.1:53 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:49837 | 192.168.1.1:53 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:64171 | 192.168.1.1:53 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:51358 | 192.168.1.1:53 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:52142 | 192.168.1.1:53 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:64911 | 192.168.1.1:53 | Normal
TCP | 27/09/2009 01:45:43 | 192.168.1.2:1447 | 74.125.19.103:80 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:53787 | 192.168.1.1:53 | Normal
TCP | 27/09/2009 01:45:39 | 192.168.1.2:4491 | 74.125.67.19:443 | Attack
TCP | 27/09/2009 01:45:39 | 192.168.1.2:1449 | 74.125.67.100:80 | Normal
UDP | 27/09/2009 01:45:44 | 192.168.1.2:58935 | 192.168.1.1:53 | Normal
TCP | 27/09/2009 01:45:42 | 192.168.1.2:1352 | 74.125.67.83:443 | Attack
TCP | 27/09/2009 01:45:44 | 192.168.1.2:1451 | 74.125.157.101:80 | Normal
```

Gambar 5.9 Potongan hasil deteksi paket data normal dan paket data berupa intrusi

5.2.1.8 Uji Coba Proses Incremental Learning

Uji coba incremental learning dilakukan dengan cara memanggil fungsi incrementalLearning pada fungsi utama program dengan parameter protokol paket data, *array of double*

ngram paket data baru, dan port tujuan paket data. Pada Tabel 5.8 merupakan prosedur pengujian lengkap yang akan dilakukan.

Tabel 5.8 Prosedur proses incremental learning

| | |
|------------------------|---|
| ID | UJ-08 |
| Nama | Uji coba incremental learning |
| Tujuan Uji Coba | Menguji apakah sistem dapat memperbaharui nilai rata-rata dan standar deviasi model jika terdapat data training baru. |
| Kondisi Awal | Aplikasi berjalan, fungsi incremental learning dimulai |
| Skenario | Memberi masukan berupa paket data normal |
| Masukan | N-gram paket data normal |
| Keluaran | Rata-rata dan standar deviasi model yang baru |
| Hasil Uji Coba | Berhasil |

Jumlah setiap variabel sebelum proses incremental learning:

[88.0, 88.0, 210.0, 39.0, 16.0, 16.0, 24.0, 5.0, 51.0, 137.0, 143.0, 90.0, 136.0, 167.0, 140.0, 117.0, 95.0, 155.0, 78.0, 84.0, 146.0, 89.0, 112.0, 77.0, 160.0, 9.0, 186.0, 26.0, 128.0, 184.0, 68.0, 21.0, 45.0, 88.0, 99.0, 60.0, 51.0, 54.0, 59.0, 57.0, 51.0, 72.0, 45.0, 47.0, 125.0, 74.0, 98.0, 56.0, 54.0, 27.0, 13.0, 33.0, 69.0, 158.0, 61.0, 202.0, 60.0, 348.0, 119.0, 114.0, 82.0, 169.0, 78.0, 85.0, 144.0, 56.0, 172.0, 209.0, 123.0, 86.0, 120.0, 110.0, 183.0, 40.0, 73.0, 134.0, 71.0, 47.0, 72.0]

Jumlah kuadrat setiap variabel sebelum proses incremental learning:

[2178.0, 2178.0, 12326.0, 513.0, 72.0, 72.0, 162.0, 11.0, 791.0, 5271.0, 5681.0, 2254.0, 5118.0, 7709.0, 5326.0, 3765.0, 2503.0, 6693.0, 1658.0, 1946.0, 5882.0, 2211.0, 3528.0, 1637.0, 7066.0, 23.0, 9380.0, 178.0, 4578.0, 2982.0, 1242.0, 125.0, 549.0, 2178.0, 2693.0, 962.0, 713.0, 830.0, 943.0, 887.0, 701.0, 1458.0, 533.0, 625.0, 4301.0, 1514.0, 2612.0, 882.0, 788.0, 189.0, 45.0, 305.0, 1345.0, 6946.0, 1833.0, 11510.0, 974.0, 33806.0, 3883.0, 3574.0, 1842.0, 7893.0, 1710.0, 2029.0, 5762.0, 838.0, 8344.0, 12139.0, 4257.0, 2038.0, 3978.0, 3404.0, 9455.0, 454.0, 1453.0, 4986.0, 1397.0, 617.0, 1442.0]

Rata-rata setiap variabel sebelum proses incremental learning:

[22.0, 22.0, 52.5, 9.75, 4.0, 4.0, 6.0, 1.25, 12.75, 34.25, 35.75, 22.5, 34.0, 41.75, 35.0, 29.25, 23.75, 38.75, 19.5, 21.0, 36.5, 22.25, 28.0, 19.25, 40.0, 2.25, 46.5, 6.5, 32.0, 26.0, 17.0, 5.25, 11.25, 22.0, 24.75, 15.0, 12.75, 13.5, 14.75, 14.25, 12.75, 18.0, 11.25, 11.75, 31.25, 18.5, 24.5, 14.0, 13.5, 6.75, 3.25, 8.25, 17.25, 39.5, 15.25, 50.5, 15.0, 87.0, 29.75, 28.5, 20.5, 42.25, 19.5, 21.25, 36.0, 14.0, 43.0, 52.25, 30.75, 21.5, 30.0, 27.5, 45.75, 10.0, 18.25, 33.5, 17.75, 11.75, 18.0]

Standar Deviasi setiap variabel sebelum proses incremental learning:

[8.98, 8.98, 20.82, 6.65, 1.63, 1.63, 2.44, 1.25, 6.84, 13.88, 13.76, 8.73, 12.83, 15.67, 11.91, 10.68, 9.06, 15.12, 6.75, 7.78, 13.57, 8.77, 11.43, 7.18, 14.89, 0.95, 15.60, 1.73, 12.67, 9.62, 5.35, 2.21, 3.77, 8.98, 8.99, 4.54, 4.57, 5.80, 4.92, 4.99, 4.11, 7.34, 2.98, 4.92, 11.47, 6.95, 8.38, 5.71, 4.43, 1.5, 0.95, 3.30, 7.18, 15.32, 5.85, 20.88, 4.96, 34.30, 10.68, 10.40, 7.32, 15.84, 7.93, 8.61, 13.88, 4.24, 17.77, 20.18, 12.57, 7.93, 11.22, 11.23, 18.99, 4.24, 6.34, 12.87, 6.75, 4.64, 6.97]

Gambar 5.10 Potongan hasil data sebelum proses incremental learning

Pada Gambar 5.10 ditunjukkan potongan hasil data sebelum proses incremental learning. Sedangkan pada Gambar 5.11 merupakan potongan hasil data setelah proses incremental

learning. Supaya hasil proses *incremental learning* dapat ditampilkan untuk memastikan nilai jumlah variable yang ditandai dengan warna merah, jumlah kuadrat variable yang ditandai dengan warna biru, rata-rata variable yang ditandai dengan warna hijau, dan standar deviasi variabel yang diwarnai dengan warna oranye diperbaharui atau tidak jika mendeteksi paket data normal maka akan dilakukan modifikasi. Modifikasi yang dilakukan adalah menambahkan perintah cetak.

```
Jumlah setiap variabel setelah proses incremental learning:
[110.0, 110.0, 262.0, 43.0, 20.0, 20.0, 30.0, 5.0, 61.0, 171.0, 177.0, 112.0, 168.0, 207.0, 171.0,
145.0, 117.0, 193.0, 96.0, 105.0, 180.0, 109.0, 140.0, 95.0, 192.0, 11.0, 230.0, 32.0, 160.0, 130.0,
84.0, 25.0, 53.0, 110.0, 123.0, 74.0, 63.0, 68.0, 73.0, 71.0, 63.0, 90.0, 55.0, 59.0, 155.0, 92.0,
122.0, 70.0, 66.0, 33.0, 15.0, 41.0, 85.0, 191.0, 75.0, 252.0, 74.0, 429.0, 147.0, 140.0, 99.0, 209.0,
98.0, 105.0, 178.0, 68.0, 212.0, 257.0, 153.0, 104.0, 143.0, 135.0, 225.0, 48.0, 91.0, 166.0, 87.0,
59.0, 90.0]

Jumlah kuadrat setiap variabel setelah proses incremental learning:
[2662.0, 2662.0, 15030.0, 529.0, 88.0, 88.0, 198.0, 11.0, 891.0, 6427.0, 6837.0, 2738.0, 6142.0,
9309.0, 6287.0, 4549.0, 2987.0, 8137.0, 1982.0, 2387.0, 7038.0, 2611.0, 4312.0, 1961.0, 8090.0, 27.0,
11316.0, 214.0, 5602.0, 3658.0, 1498.0, 141.0, 613.0, 2662.0, 3269.0, 1158.0, 857.0, 1026.0, 1139.0,
1083.0, 845.0, 1782.0, 633.0, 769.0, 5201.0, 1838.0, 3188.0, 1078.0, 932.0, 225.0, 49.0, 369.0, 1601.0,
8035.0, 1229.0, 14010.0, 1170.0, 40367.0, 4667.0, 4250.0, 2131.0, 9493.0, 2110.0, 2429.0, 6918.0,
982.0, 9944.0, 14443.0, 5157.0, 2362.0, 4507.0, 4029.0, 11219.0, 518.0, 1777.0, 6010.0, 1653.0, 761.0,
1766.0]

Rata-rata setiap variabel setelah proses incremental learning:
[22.0, 22.0, 52.4, 8.6, 4.0, 4.0, 6.0, 1.0, 12.2, 34.2, 35.4, 22.4, 33.6, 41.4, 34.2, 29.0, 23.4, 38.6,
19.2, 21.0, 36.0, 21.8, 28.0, 19.0, 38.4, 2.2, 46.0, 6.4, 32.0, 26.0, 16.8, 5.0, 10.6, 22.0, 24.6,
14.8, 12.6, 13.6, 14.6, 14.2, 12.6, 18.0, 11.0, 11.8, 31.0, 18.4, 24.4, 14.0, 13.2, 6.6, 3.0, 8.2,
17.0, 38.2, 15.0, 50.4, 14.8, 85.8, 29.4, 28.0, 19.8, 41.8, 19.6, 21.0, 35.6, 13.6, 42.4, 51.4, 30.6,
20.8, 28.6, 27.0, 45.0, 9.6, 18.2, 33.2, 17.4, 11.8, 18.0]

Standar Deviasi setiap variabel setelah proses incremental learning:
[7.77, 7.78, 18.03, 6.30, 1.41, 1.41, 2.12, 1.22, 6.05, 12.02, 11.94, 7.56, 11.48, 13.59, 10.47, 9.27,
7.89, 13.10, 5.89, 6.74, 11.81, 7.66, 9.89, 6.24, 13.39, 0.83, 13.56, 1.51, 10.97, 8.33, 4.65, 2.0,
3.57, 7.78, 7.79, 3.96, 3.97, 5.02, 4.27, 4.32, 3.57, 6.36, 2.64, 4.26, 9.94, 6.02, 7.26, 4.94, 3.89,
1.34, 1.0, 2.86, 6.24, 13.59, 5.09, 18.09, 4.32, 29.82, 9.28, 9.08, 6.53, 13.75, 6.87, 7.48, 12.05,
3.78, 15.45, 17.55, 10.89, 7.04, 10.21, 9.79, 16.53, 3.78, 5.49, 11.16, 5.89, 4.02, 6.04]
```

Gambar 5.11 Potongan hasil data setelah proses incremental learning

5.2.2 Uji Coba Performa

Uji coba performa dilakukan untuk mengetahui kemampuan dari sistem dalam mendeteksi serangan serta pengaruhnya terhadap kondisi komputer yang menjalankan sistem ini terdapat 2 bagian dalam uji coba performa yaitu:

- Uji coba performa sistem secara umum, meliputi: utilisasi CPU, utilisasi RAM, perbandingan kualitas akses ke server ketika kondisi normal dan terjadi serangan, serta kecepatan pendeteksian

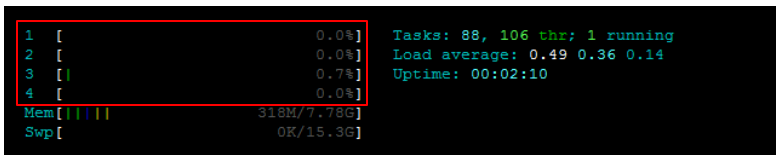
- Uji coba akurasi dengan pengolahan data uji menggunakan metode *two fold cross validation* dan perhitungan validasi menggunakan metode *confussion matrix*. Uji coba ini dilakukan untuk mengetahui lebih dalam mengenai tingkat akurasi dan presisi dari sistem dan metode Mahalanobis Distance.

5.2.2.1 Uji Coba Performa Sistem

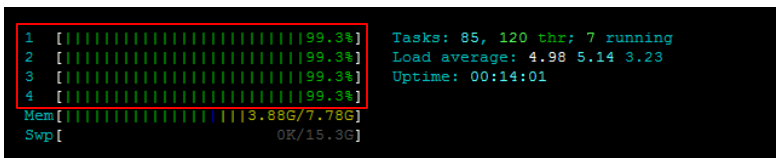
Uji coba utilisasi pada komputer dilakukan untuk mengetahui seberapa besar sistem yang dijalankan mempengaruhi performa dari komputer tempat sistem ini terpasang.

1. Utilisasi CPU

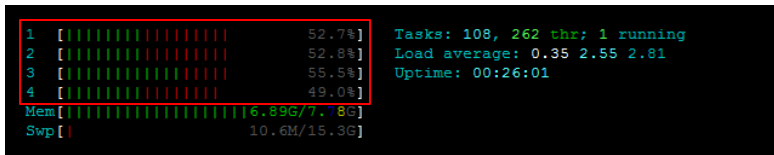
Uji coba utilisasi CPU dilakukan dengan cara membandingkan persentasi utilisasi CPU ketika sistem tidak dijalankan dan ketika *training* dan identifikasi dijalankan. Aplikasi HTOP digunakan untuk membantu mengetahui utilisasi CPU. Gambar 5.12 hingga Gambar 5.14 menunjukkan hasil dari pengujian.



Gambar 5.12 HTOP CPU ketika sistem belum berjalan

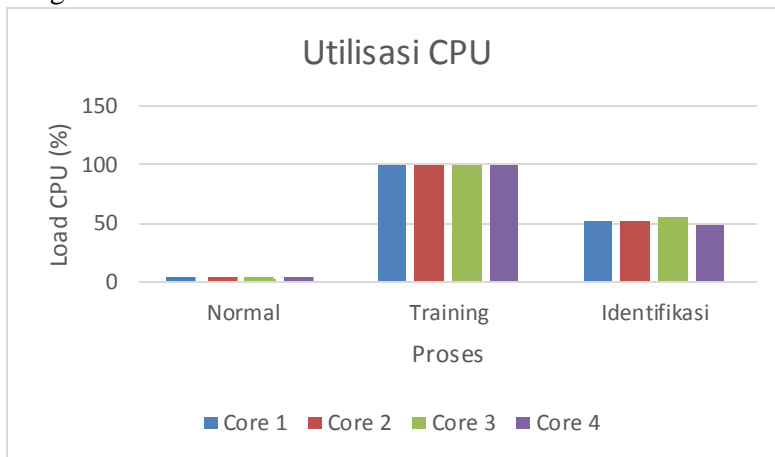


Gambar 5.13 HTOP CPU ketika *training* data set berjalan



Gambar 5.14 HTOP CPU ketika identifikasi berjalan

Berdasarkan pada Gambar 5.12 hingga Gambar 5.14, terlihat bahwa terjadi peningkatan utilisasi CPU ketika proses *training* data set berjalan dan proses identifikasi berjalan. Pada Gambar 5.15 akan menyajikan tingkat peningkatan utilisasi CPU dengan lebih detail.



Gambar 5.15 Grafik persentase utilisasi CPU

Berdasarkan grafik pada Gambar 5.15 terlihat bahwa peningkatan persentase utilisasi CPU sangat signifikan ketika proses *training* data set dan proses identifikasi terjadi. Utilisai CPU paling tinggi terjadi pada proses training data set, karena pada proses tersebut sistem membaca file data set dengan menggunakan *thread* sejumlah file data set.

2. Utilisasi RAM

Uji coba utilisasi RAM dilakukan dengan cara membandingkan persentasi utilisasi RAM ketika sistem tidak dijalankan dan ketika *training* dan identifikasi dijalankan. Aplikasi HTOP digunakan untuk membantu mengetahui utilisasi RAM. Pada Gambar 5.16 hingga Gambar 5.18 menunjukkan hasil dari pengujian.

```

1 [          0.0%] Tasks: 88, 106 thr; 1 running
2 [          0.0%] Load average: 0.49 0.36 0.14
3 [          0.7%] Uptime: 00:02:10
4 [          0.0%]
Mem[|||||          318M/7.78G]
Swp[          0K/15.3G]

```

Gambar 5.16 HTOP RAM ketika sistem belum berjalan

```

1 [|||||99.3%] Tasks: 85, 120 thr; 7 running
2 [|||||99.3%] Load average: 4.98 5.14 3.23
3 [|||||99.3%] Uptime: 00:14:01
4 [|||||99.3%]
Mem[|||||99.3%]
Swp[          0K/15.3G]

```

Gambar 5.17 HTOP RAM ketika *training* data set berjalan

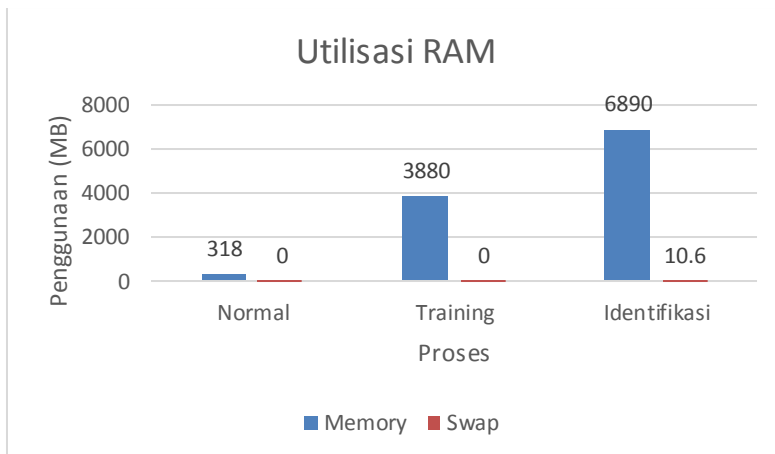
```

1 [|||||52.7%] Tasks: 108, 262 thr; 1 running
2 [|||||52.8%] Load average: 0.35 2.55 2.81
3 [|||||55.5%] Uptime: 00:26:01
4 [|||||49.0%]
Mem[|||||6.89G/7.78G]
Swp[          10.6M/15.3G]

```

Gambar 5.18 HTOP RAM ketika identifikasi berjalan

Berdasarkan pada Gambar 5.16 hingga Gambar 5.18, terlihat bahwa terjadi peningkatan utilisasi RAM ketika proses *training* data set berjalan. Pada Gambar 5.19 akan menyajikan peningkatan utilisasi RAM dengan lebih detail.

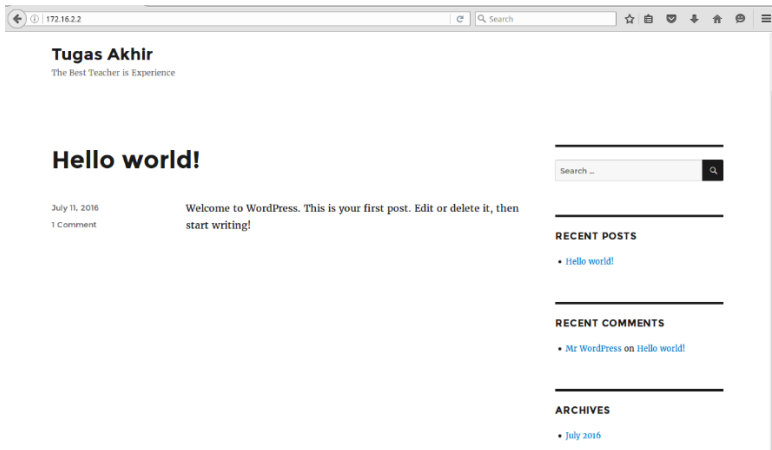


Gambar 5.19 Grafik persentase utilisasi RAM

Berdasarkan grafik pada Gambar 5.19 terlihat bahwa peningkatan persentase utilisasi RAM sangat signifikan ketika proses *training* data set dan proses identifikasi terjadi. Utilisasi RAM pada proses *training* data set dan identifikasi sangat tinggi, karena pada proses tersebut konten paket yang telah diproses menggunakan metode *n-gram* lalu disimpan pada *arraylist*, dimana *n-gram* setiap model disimpan ke dalam sebuah *arraylist* yang memiliki ukuran sepanjang 256.

3. Performa Layanan Jaringan

Uji coba performa layanan jaringan ini untuk mengetahui bagaimana kualitas jaringan ketika diakses oleh pengguna. Gambar 5.20 merupakan tampilan halaman yang akan diuji untuk diakses oleh ApacheBench. ApacheBench merupakan aplikasi yang dapat digunakan untuk mengetahui waktu yang diperlukan untuk menyelesaikan sebuah *request* ke sebuah *web server*. Dengan aplikasi ApacheBench dapat mengirimkan beberapa *request* ke sebuah *web server* dalam satu waktu.



Gambar 5.20 Tampilan halaman web yang akan diakses

Pengujian dilakukan dengan bantuan aplikasi ApacheBench yang akan mengakses halaman web yang ada pada server target dengan *concurrency* 10 dan 1000 *request*. *Concurrency* tersebut menjelaskan bahwa akan ada 10 thread yang akan berjalan untuk mengakses. Terdapat beberapa skenario yang disiapkan. Skenario tersebut disiapkan disesuaikan dengan kondisi-kondisi yang paling mungkin ditemui dalam kegiatan ini. Akan dilakukan beberapa skenario pengujian yaitu:

1. Skenario 1: ApacheBench dijalankan ketika kondisi jaringan normal dan sistem tidak dijalankan
2. Skenario 2: ApacheBench dijalankan ketika kondisi jaringan normal dan aplikasi pendeteksian dijalankan
3. Skenario 3: ApacheBench dijalankan ketika kondisi komputer target diserang dan aplikasi pendeteksian dijalankan

Gambar 5.21 hingga Gambar 5.23 memuat cuplikan data dari ApacheBench yang merupakan hasil percobaan mengakses halaman web pada server.

```

ta@bangau:~$ ab -n 1000 -c 10 http://172.16.2.2:80/ta/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.2.2 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.7
Server Hostname:      172.16.2.2
Server Port:          80

Document Path:        /ta/
Document Length:      10028 bytes

Concurrency Level:    10
Time taken for tests:  28.975 seconds
Complete requests:    1000
Failed requests:       0
Total transferred:    10304000 bytes
HTML transferred:     10028000 bytes
Requests per second:  34.51 [#/sec] (mean)
Time per request:     289.746 [ms] (mean)
Time per request:     28.975 [ms] (mean, across all concurrent requests)
Transfer rate:        347.28 [Kbytes/sec] received

Connection Times (ms)
      min   mean[+/-sd] median   max
Connect:    0    2   4.4      0    43
Processing: 134  288  20.4    287   481
Waiting:    130  239  17.1    238   392
Total:      135  289  21.5    288   481

Percentage of the requests served within a certain time (ms)
 50%    288
 66%    294
 75%    297
 80%    300
 90%    307
 95%    316
 98%    353
 99%    369
100%    481 (longest request)
ta@bangau:~$

```

Gambar 5.21 Luaran ApacheBench untuk skenario 1

Berdasarkan pada Gambar 5.21, untuk skenario 1, diperlukan 28.975 detik untuk menyelesaikan pengujian dengan rata-rata 34.51 milidetik untuk menyelesaikan 1 *request*.


```

ta@bangau:~$ ab -n 1000 -c 10 http://172.16.2.2:80/ta/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.2.2 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.7
Server Hostname:      172.16.2.2
Server Port:          80

Document Path:        /ta/
Document Length:      10028 bytes

Concurrency Level:    10
Time taken for tests:  28.779 seconds
Complete requests:    1000
Failed requests:       0
Total transferred:    10304000 bytes
HTML transferred:     10028000 bytes
Requests per second:  34.75 [#/sec] (mean)
Time per request:     287.791 [ms] (mean)
Time per request:     28.779 [ms] (mean, across all concurrent requests)
Transfer rate:        349.65 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:        0    1   1.6      0    11
Processing:    126  286  18.3    286   411
Waiting:       122  238  16.0    238   336
Total:         126  287  18.3    287   412

Percentage of the requests served within a certain time (ms)
 50%    287
 66%    293
 75%    297
 80%    299
 90%    307
 95%    313
 98%    326
 99%    334
100%    412 (longest request)
ta@bangau:~$

```

Gambar 5.22 Luaran ApacheBecnh untuk skenario 2

Berdasarkan pada Gambar 5.22, untuk skenario 1, diperlukan 28.779 detik untuk menyelesaikan pengujian dengan rata-rata 34.75 milidetik untuk menyelesaikan 1 *request*.

```

ta@bangau:~$ ab -n 1000 -c 10 http://172.16.2.2:80/ta/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.2.2 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.7
Server Hostname:      172.16.2.2
Server Port:          80

Document Path:        /ta/
Document Length:      10028 bytes

Concurrency Level:     10
Time taken for tests:  28.662 seconds
Complete requests:     1000
Failed requests:        0
Total transferred:      10304000 bytes
HTML transferred:      10028000 bytes
Requests per second:    34.89 [#/sec] (mean)
Time per request:       286.619 [ms] (mean)
Time per request:       28.662 [ms] (mean, across all concurrent requests)
Transfer rate:          351.08 [Kbytes/sec] received

Connection Times (ms)
      min  mean[+/-sd] median  max
Connect:    0    1  2.1      0    29
Processing: 134  285  17.5    285   411
Waiting:    127  237  14.9    237   349
Total:      134  286  17.6    287   412

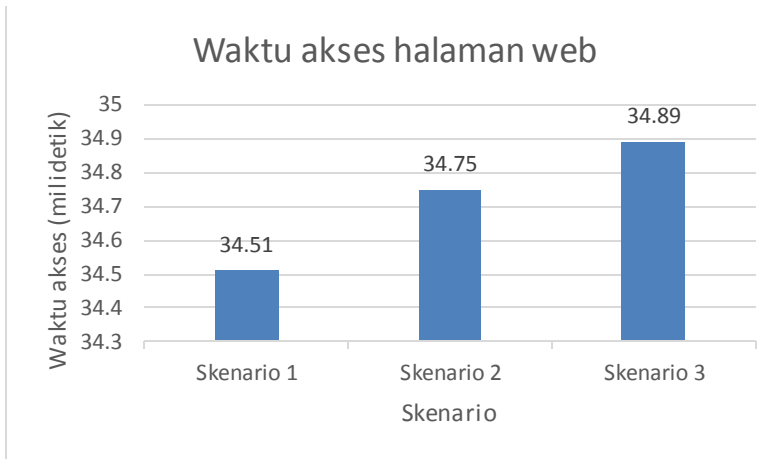
Percentage of the requests served within a certain time (ms)
 50%    287
 66%    291
 75%    294
 80%    297
 90%    304
 95%    310
 98%    322
 99%    341
100%    412 (longest request)
ta@bangau:~$

```

Gambar 5.23 Luaran ApacheBench untuk skenario 3

Berdasarkan pada Gambar 5.23, untuk skenario 1, diperlukan 28.662 detik untuk menyelesaikan pengujian dengan rata-rata 34.89 milidetik untuk menyelesaikan 1 *request*. Setelah

dilakukan pengujian dengan menggunakan 3 skenario tersebut, maka untuk lebih memudahkan visualisasi hasil pengujian akan disajikan dalam bentuk grafik pada Gambar 5.24.



Gambar 5.24 Grafik waktu akses web

5.2.2.2 Uji Coba Kecepatan Pendeteksian

Pengujian terhadap kecepatan pendeteksian dilakukan dengan menguji sistem dengan berbagai variasi besarnya ukuran *window* dari sistem. Pada uji coba performa kecepatan pendeteksian ini akan menggunakan tiga skenario, dimana pada masing-masing skenario besarnya ukuran *window* yang dilakukan berbeda-beda. Parameter lainnya sama dengan parameter ketika dilakukan uji coba fungsional. Berikut data skenario yang akan dilakukan:

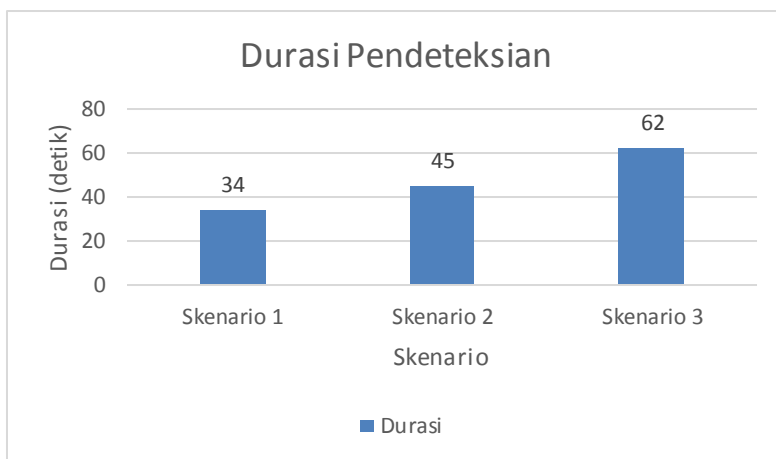
1. **Skenario 1:** *window size* 1000 paket data
2. **Skenario 2:** *window size* 1500 paket data
3. **Skenario 3:** *window size* 2000 paket data

Setelah mencoba melakukan uji coba dengan tiga skenario diatas, didapatkan data durasi waktu yang diperlukan untuk

memproses hasil *sniffing* menjadi *log* pendeteksian serangan. Pada Tabel 5.9 disajikan data pengujian dan pada Gambar 5.25 disajikan grafik performanya.

Tabel 5.9 Metode akses komputer penyerang

| No | Nama | Start | Stop | Selisih |
|----|------------|-------|-------|----------|
| 1 | Skenario 1 | 28:10 | 28:44 | 34 detik |
| 2 | Skenario 2 | 30:12 | 30:57 | 45 detik |
| 3 | Skenario 3 | 32:08 | 33:10 | 62 detik |



Gambar 5.25 Grafik durasi waktu pendeteksian serangan

5.2.2.3 Uji Coba Akurasi

Uji coba akurasi dilakukan dengan pertama-tama menyiapkan data uji yang akan digunakan. Data uji yang nantinya akan diproses dengan metode *two fold crossvalidation* [8]. Metode pengujian ini digunakan karena sistem yang dibuat memiliki sifat mirip dengan aplikasi-aplikasi yang berbasis *machine learning*. Tahap *learning* yang dimaksud pada sistem ini adalah tahap untuk penentuan *threshold* pendeteksian. Berangkat dari hal ini, maka metode *two fold cross validation* dipilih sehingga kebutuhan untuk

mendapatkan *threshold* dan data uji terpenuhi. Dengan metode *two fold cross validation*, kumpulan data uji akan dibagi 2 sama banyak. Kemudian salah satu kumpulan data akan dijadikan data *training* sekaligus menentukan *threshold* dari aplikasi pendeteksian. Setelah selesai, maka kegiatan diulang kembali dengan kumpulan data yang sebelumnya menjadi data uji akan dijadikan data *training* dan kumpulan data *training* dijadikan data uji. Berikut pada Tabel 5.10 disajikan data uji dimana parameter pembeda antar data uji adalah besarnya ukuran *window*. *Window size* yang dimaksud adalah jumlah paket data yang ditangkap, jika jumlah paket data yang ditangkap sudah memenuhi *window size* baru paket data tersebut diproses untuk mengetahui jarak mahalanobis setiap paket. Data uji yang digunakan adalah *file* paket data hasil tangkapan paket data pada jaringan eksternal DARPA pada minggu ke-4 yang berjumlah 5 *file* paket data.

Data yang akan dicatat nantinya adalah jarak mahalanobis dari paket data normal dan paket data yang berupa intrusi. Kedua variable tersebut nantinya akan diolah menjadi nilai *threshold* dengan cara menjumlahkan jarak mahalanobis terkecil dari paket data yang berupa intrusi dengan jarak mahalanobis terbesar dari paket data normal dan kemudian dibagi 2. Berikut pada persamaan (5.1) dijabarkan cara untuk mendapatkan nilai *threshold* pendeteksian.

$$Threshold = \frac{\min \text{ jarak serangan} + \max \text{ jarak normal}}{2} \quad (5.1)$$

Tabel 5.10 Data uji

| No | Ukuran <i>window</i> | Port TCP | | | | Port UDP |
|----|----------------------|----------|----|----|----|----------|
| 1 | 10000 | 21 | 23 | 25 | 80 | 53 |
| 2 | 15000 | 21 | 23 | 25 | 80 | 53 |
| 3 | 20000 | 21 | 23 | 25 | 80 | 53 |

Untuk menghitung akurasi, digunakan metode *confussion matrix* [9]. Penggunaan metode ini digunakan karena tergolong

mudah untuk digunakan dan dapat menghasilkan nilai-nilai pengujian selain akurasi, seperti *true positive rate* dan *false positive rate*. Dalam perhitungan akurasi yang menggunakan metode *confussion matrix*, selain data uji serangan predeksi benar diperlukan juga data uji serangan untuk prediksi salah. Untuk itu maka dibuat juga data uji prediksi salah dimana tidak ada serangan dan hanya ada akses pengguna biasa. Data uji ini hanya digunakan ketika fase pengujian dan tidak digunakan ketika fase *training* hal ini dikarenakan fase *training* digunakan untuk menentukan *threshold* jarak mahalanobis serangan, sehingga dibutuhkan data uji yang mengandung serangan. *Confussion matrix* yang akan digunakan untuk pengujian akurasi adalah *confussion matrix* dengan ukuran 2x2. Berikut pada Gambar 5.26 ditunjukkan model *confussion matrix* beserta kemudian dijelaskan definisi masing-masing kelasnya.

| | | PREDIKSI | |
|----------|-------|----------|-------|
| | | TRUE | FALSE |
| REALITAS | TRUE | A | B |
| | FALSE | C | D |

Gambar 5.26 Model Confussion Matrix untuk pengujian

Berikut penjelasan dari masing-masing kelas pada matrix:

- Kelas A: Kondisi ketika data prediksi benar, ketika diuji juga menghasilkan luaran yang benar
- Kelas B: Kondisi ketika data prediksi salah, ketika diuji menghasilkan luaran benar
- Kelas C: Kondisi ketika data prediksi benar, ketika diuji menghasilkan luaran salah

- Kelas D: Kondisi ketika data prediksi salah, ketika diuji juga menghasilkan luaran salah
Perhitungan yang akan dilakukan berdasarkan klasifikasi pada *confussion matrix* adalah sebagai berikut:

- Akurasi (AC) adalah perhitungan untuk total prediksi yang benar. Nilai akurasi bisa didapat dengan rumus pada persamaan (5.2).

$$AC = \frac{A + D}{A + B + C + D} \quad (5.2)$$

- *True positive rate* (TP) adalah perhitungan proporsi dari kasus benar yang diidentifikasi dengan benar. Nilai akurasi bisa didapat dengan rumus pada persamaan (5.3).

$$TP = \frac{A}{A + B} \quad (5.3)$$

- *False positive rate* (FP) adalah proporsi dari kasus salah yang diidentifikasi sebagai data benar. Nilai akurasi bisa didapat dengan rumus pada persamaan (5.4).

$$FP = \frac{C}{C + D} \quad (5.4)$$

- *True negative rate* (TN) adalah proporsi dari kasus salah yang diidentifikasi dengan benar sebagai data salah. Nilai TN bisa didapat dengan rumus pada persamaan (5.5).

$$TN = \frac{D}{C + D} \quad (5.5)$$

- *False negative rate* (FN) adalah proporsi dari kasus benar yang diidentifikasi sebagai data salah. Nilai FN bisa didapat dengan rumus pada persamaan (5.6).

$$FN = \frac{B}{A + B} \quad (5.6)$$

- Presisi (P) adalah proporsi dari prediksi positif yang diidentifikasi dengan benar. Nilai dari P bisa didapat dengan rumus pada persamaan (5.7).

$$P = \frac{A}{A + C} \quad (5.7)$$

1. Pengujian 1

Untuk percobaan 1, data *testing* dari DARPA minggu ke-4, hari ke-1 akan digunakan sebagai data *training* untuk menetapkan *threshold* dari sistem. Kemudian, data *testing* minggu ke-5, hari ke-1 akan dijadikan data untuk pengujian.

Berikut pada Tabel 5.11 hingga Tabel 5.12 merupakan hasil pengujian dengan menggunakan data *training* minggu ke-4, hari ke-1.

Tabel 5.11 Hasil Uji Data *Training* minimum jarak paket data intrusi

| No | Window size | Jarak Port TCP | | | | Jarak Port UDP |
|----|-------------|----------------|-------|-------|------|----------------|
| | | 21 | 23 | 25 | 80 | 53 |
| 1 | 10000 | 26.46 | 25.66 | 13.51 | 0.00 | 3.44 |
| 2 | 15000 | 26.46 | 22.52 | 13.51 | 0.00 | 5.41 |
| 3 | 20000 | 26.46 | 22.52 | 12.3 | 0.00 | 6.7 |

Tabel 5.12 Hasil Uji Data *Training* maksimum jarak paket data normal

| No | Window size | Jarak Port TCP | | | | Jarak Port UDP |
|----|-------------|----------------|-------|--------|------|----------------|
| | | 21 | 23 | 25 | 80 | 53 |
| 1 | 10000 | 80.13 | 0 | 426.68 | 0.00 | 156.91 |
| 2 | 15000 | 80.13 | 0 | 426.68 | 0.00 | 210.08 |
| 3 | 20000 | 80.13 | 29.06 | 426.68 | 0.00 | 253.59 |

Berdasarkan data pada Tabel 5.11 dan Tabel 5.12 maka dapat diambil data untuk menentukan *threshold* pada masing-masing port yaitu mengambil data yang memiliki jarak paling kecil

untuk paket data intrusi dan memiliki jarak paling besar untuk paket data normal. Jika kedua data tersebut diolah dengan menggunakan persamaan (5.1) maka didapatkan threshold untuk masing-masing port yang disajikan pada Tabel 5.13.

Tabel 5.13 Threshold untuk masing-masing port

| No | Threshold | | | | |
|----|----------------|-------|--------|------|----------|
| | Jarak Port TCP | | | | Port UDP |
| | 21 | 23 | 25 | 80 | 53 |
| 1 | 53.30 | 25.79 | 219.49 | 0.00 | 128.52 |

1.1. Pengujian 1a

Dengan berbekal *threshold* tersebut maka dilakukan pengujian terhadap data testing minggu ke-5, hari ke-1 tanpa proses *incremental learning*. Hasil dari pengujian dituliskan pada Tabel 5.14.

Tabel 5.14 Hasil Uji Data Testing minggu ke-5 tanpa proses *incremental learning*

| No | Window size | Jumlah connection | Jumlah Paket normal | Jumlah Paket serangan |
|----|-------------|-------------------|---------------------|-----------------------|
| 1 | 10000 | 5328 | 5308 | 20 |
| 2 | 20000 | 8515 | 8478 | 37 |

Setelah semua data diuji, maka data dapat diproses hasilnya dengan *confussion matrix*. Pada Tabel 5.15 disajikan klasifikasi jumlah masing-masing kelas berdasarkan pada hasil pengujian.

Tabel 5.15 Confussion matrix uji coba 1a

| No | Window size | Kelas | | | |
|----|-------------|-------|-----|----|------|
| | | A | B | C | D |
| 1 | 10000 | 6 | 355 | 14 | 4935 |
| 2 | 20000 | 15 | 552 | 22 | 7926 |

Berdasarkan pada jumlah diatas, maka didapatkan penilaian berdasarkan rumus-rumus yang terkait dengan

confussion matix yang disajikan pada Tabel 5.16 hingga Tabel 5.17.

Tabel 5.16 Hasil penilaian percobaan dengan ukuran window 10000

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|--------|------------|
| 1 | Akurasi (AC) | 0.9307 | 93.07% |
| 2 | <i>True positive rate</i> (TP) | 0.0166 | 1.66% |
| 3 | <i>False negative rate</i> (FN) | 0.9834 | 98.34% |
| 4 | <i>False positive rate</i> (FP) | 0.0028 | 0.28% |
| 5 | <i>True negative rate</i> (TN) | 0.9972 | 99.72% |
| 6 | Presisi (P) | 0.3 | 30.0% |

Tabel 5.17 hasil penilaian percobaan dengan ukuran window 20000

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|--------|------------|
| 1 | Akurasi (AC) | 0.9324 | 93.26% |
| 2 | <i>True positive rate</i> (TP) | 0.0265 | 2.65% |
| 3 | <i>False negative rate</i> (FN) | 0.9375 | 97.35% |
| 4 | <i>False positive rate</i> (FP) | 0.0028 | 0.28% |
| 5 | <i>True negative rate</i> (TN) | 0.9972 | 99.82% |
| 6 | Presisi (P) | 0.4054 | 40.54% |

1.2. Pengujian 1b

Dengan berbekal *threshold* tersebut maka dilakukan pengujian terhadap data testing minggu ke-5, hari ke-1 dengan tambahan proses *incremental learning*. Hasil dari pengujian dituliskan pada Tabel 5.18

Tabel 5.18 Hasil Uji Data Testing minggu ke-5 dengan proses *incremental learning*

| No | Window size | Jumlah connection | Jumlah Paket normal | Jumlah Paket serangan |
|----|-------------|-------------------|---------------------|-----------------------|
| 1 | 10000 | 5328 | 1577 | 3751 |
| 2 | 20000 | 8515 | 6711 | 1804 |

Setelah semua data diuji, maka data dapat diproses hasilnya dengan *confussion matrix*. Pada Tabel 5.19 disajikan klasifikasi jumlah masing-masing kelas berdasarkan pada hasil pengujian.

Tabel 5.19 Confussion matrix uji coba 1b

| No | Window size | Kelas | | | |
|----|-------------|-------|-----|------|------|
| | | A | B | C | D |
| 1 | 10000 | 187 | 174 | 3564 | 1403 |
| 2 | 20000 | 314 | 253 | 6397 | 1551 |

Berdasarkan pada jumlah diatas, maka didapatkan penilaian berdasarkan rumus-rumus yang terkait dengan *confussion matix* yang disajikan pada Tabel 5.20 hingga Tabel 5.21

Tabel 5.20 Hasil penilaian percobaan dengan ukuran window 10000

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|--------|------------|
| 1 | Akurasi (AC) | 0.2984 | 29.84% |
| 2 | <i>True positive rate</i> (TP) | 0.518 | 51.58% |
| 3 | <i>False negative rate</i> (FN) | 0.482 | 48.2% |
| 4 | <i>False positive rate</i> (FP) | 0.7175 | 71.75% |
| 5 | <i>True negative rate</i> (TN) | 0.2825 | 28.25% |
| 6 | Presisi (P) | 0.0499 | 4.99% |

Tabel 5.21 hasil penilaian percobaan dengan ukuran window 20000

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|--------|------------|
| 1 | Akurasi (AC) | 0.219 | 21.9% |
| 2 | <i>True positive rate</i> (TP) | 0.5538 | 55.38% |
| 3 | <i>False negative rate</i> (FN) | 0.4462 | 44.62% |
| 4 | <i>False positive rate</i> (FP) | 0.8049 | 80.49% |
| 5 | <i>True negative rate</i> (TN) | 0.1951 | 19.51% |
| 6 | Presisi (P) | 0.0468 | 4.68% |

2. Pengujian 2

Untuk percobaan 2, data *testing* dari DARPA minggu ke-4 akan digunakan sebagai data *training* untuk menetapkan *threshold* dari sistem. Kemudian, untuk data pengujian didapat dari dengan mengirimkan paket intrusi langsung ke komputer target.

2.1. Pengujian 2a

Dengan berbekal *threshold* tersebut maka dilakukan pengujian terhadap data testing dengan skenario pada Tabel 5.22 tanpa proses *incremental learning*. Hasil dari pengujian dituliskan pada Tabel 5.23.

Tabel 5.22 Skenario serangan

| No | Jenis serangan | Jumlah paket |
|----|--------------------------|--------------|
| 1 | FTP login brute force | 1000 |
| 2 | Telnet login brute force | 1000 |

Tabel 5.23 Hasil Uji Data Testing secara real-time

| No | Window size | Jumlah connection | Jumlah Paket normal | Jumlah Paket serangan |
|----|-------------|-------------------|---------------------|-----------------------|
| 1 | 1000 | 151 | 0 | 151 |
| 2 | 1000 | 290 | 0 | 290 |

Setelah semua data diuji, maka data dapat diproses hasilnya dengan *confussion matrix*. Berikut klasifikasi jumlah masing-masing kelas berdasarkan pada hasil pengujian.

Tabel 5.24 Confussion matrix uji coba 2a

| No | Serangan | Window size | Kelas | | | |
|----|--------------------------|-------------|-------|---|---|---|
| | | | A | B | C | D |
| 1 | FTP login brute force | 1000 | 151 | 0 | 0 | 0 |
| 2 | Telnet login brute force | 1000 | 290 | 0 | 0 | 0 |

Berdasarkan pada jumlah diatas, maka didapatkan penilaian berdasarkan rumus-rumus yang terkait dengan *confussion matix* yang disajikan pada Tabel 5.25 hingga Tabel 5.26.

Tabel 5.25 Hasil penilaian percobaan FTP brute force

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|-------|------------|
| 1 | Akurasi (AC) | 1 | 100 % |
| 2 | <i>True positive rate</i> (TP) | 0 | 0% |
| 3 | <i>False negative rate</i> (FN) | 0 | 0% |
| 4 | <i>False positive rate</i> (FP) | 0 | 0% |
| 5 | <i>True negative rate</i> (TN) | 0 | 0% |
| 6 | Presisi (P) | 0 | 0% |

Tabel 5.26 Hasil penilaian percobaan Telnet brute force

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|-------|------------|
| 1 | Akurasi (AC) | 1 | 100 % |
| 2 | <i>True positive rate</i> (TP) | 0 | 0% |
| 3 | <i>False negative rate</i> (FN) | 0 | 0% |
| 4 | <i>False positive rate</i> (FP) | 0 | 0% |
| 5 | <i>True negative rate</i> (TN) | 0 | 0% |
| 6 | Presisi (P) | 0 | 0% |

2.2. Pengujian 2b

Dengan berbekal *threshold* tersebut maka dilakukan pengujian terhadap data testing dengan skenario pada Tabel 5.22 tanpa proses *incremental learning*. Hasil dari pengujian dituliskan pada Tabel 5.23.

Tabel 5.27 Skenario serangan

| No | Jenis serangan | Jumlah paket |
|----|--------------------------|--------------|
| 1 | FTP login brute force | 1000 |
| 2 | Telnet login brute force | 1000 |

Tabel 5.28 Hasil Uji Data Testing secara real-time

| No | Window size | Jumlah connection | Jumlah Paket normal | Jumlah Paket Serangan |
|----|-------------|-------------------|---------------------|-----------------------|
| 1 | 1000 | 159 | 0 | 159 |
| 2 | 2000 | 290 | 0 | 290 |

Setelah semua data diuji, maka data dapat diproses hasilnya dengan *confussion matrix*. Berikut klasifikasi jumlah masing-masing kelas berdasarkan pada hasil pengujian.

Tabel 5.29 Confussion matrix uji coba 2b

| No | Serangan | Window size | Kelas | | | |
|----|--------------------------|-------------|-------|---|---|---|
| | | | A | B | C | D |
| 1 | FTP login brute force | 1000 | 151 | 0 | 0 | 0 |
| 2 | Telnet login brute force | 1000 | 290 | 0 | 0 | 0 |

Berdasarkan pada jumlah diatas, maka didapatkan penilaian berdasarkan rumus-rumus yang terkait dengan *confussion matix* yang disajikan pada Tabel 5.30 hingga Tabel 5.31.

Tabel 5.30 Hasil penilaian percobaan FTP brute force

| No | Jenis Penilaian | Nilai | Persentase |
|----|---------------------------------|-------|------------|
| 1 | Akurasi (AC) | 1 | 100 % |
| 2 | <i>True positive rate</i> (TP) | 0 | 0% |
| 3 | <i>False negative rate</i> (FN) | 0 | 0% |
| 4 | <i>False positive rate</i> (FP) | 0 | 0% |
| 5 | <i>True negative rate</i> (TN) | 0 | 0% |
| 6 | Presisi (P) | 0 | 0% |

Tabel 5.31 Hasil penilaian percobaan Telnnet *brute force*

| No | Jenis Penilaian | Nilai | Persentase |
|-----------|---------------------------------|--------------|-------------------|
| 1 | Akurasi (AC) | 1 | 100 % |
| 2 | <i>True positive rate</i> (TP) | 0 | 0% |
| 3 | <i>False negative rate</i> (FN) | 0 | 0% |
| 4 | <i>False positive rate</i> (FP) | 0 | 0% |
| 5 | <i>True negative rate</i> (TN) | 0 | 0% |
| 6 | Presisi (P) | 0 | 0% |

[Halaman ini sengaja dikosongkan]

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. Dapat membangun sebuah sistem deteksi intrusi yang dapat membaca data set dari DARPA serta dapat menangkap paket data dari *network interface* secara *real-time*.
2. Metode Mahalanobis *Distance* dapat mengklasifikasikan antara paket data normal dan paket data yang berupa intrusi, namun tidak berlaku untuk protokol HTTP.
3. Sistem yang dibuat untuk pendeteksi serangan menggunakan metode Mahalanobis *Distance* tanpa proses *incremental learning* dapat mendeteksi serangan dengan persentase kebenaran 93%, dan dengan tambahan proses *incremental learning* dapat mendeteksi serangan dengan persentase kebenaran 20%.

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 *distanc* dengan proses *incremental learning* kurang akurat dibandingkan tanpa proses *incremental leraning*. Perlu ada

implementasi metode lain sehingga dapat membantu meningkatkan keakuratan pendeteksian serangan.

DAFTAR PUSTAKA

- [1] SANS Institute, "Understanding Intrusion Detection System," *SANS Institute Reading Room*, pp. 1-9, 2001.
- [2] "Sistem deteksi intrusi," [Online]. Available: https://id.wikipedia.org/wiki/Sistem_deteksi_intrusi. [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/CST_corpora/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] "Intrusion detection system," [Online]. Available: https://en.wikipedia.org/wiki/Intrusion_detection_system. [Diakses 22 June 2016].

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Bagian ini merupakan lampiran 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.getDeviiasiData(), 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.getDevisasiData(), 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");
    }
}

```


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.setDeviassiData(deviassiNew);
        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);
}
}
}

```

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 kuliah 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.