TUGAS AKHIR – KI141502

APLIKASI PENDETEKSI SERANGAN PADA HTTP MENGGUNAKAN N-GRAM

Reinhard Ruben R
NRP 5111100110

Dosen Pembimbing
Henning TiTi Ciptaningtyas, S.Kom., M.Kom.
Bagus Jati Santoso, S.Kom, Ph.D

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

TUGAS AKHIR – KI 141502

APLIKASI PENDETEKSI SERANGAN PADA HTTP MENGGUNAKAN N-GRAM

Reinhard Ruben R.
NRP 5111100110

Dosen Pembimbing
Henning TiTi Ciptaningtyas, S.Kom., M.Kom.
Bagus Jati Santoso, S.Kom, Ph.D

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

(Halaman Ini Sengaja dikosongkan)

FINAL PROJECT– KI 141502

APPLICATION FOR HTTP ATTACK DETECTION USING N-GRAM

Reinhard Ruben R.
NRP 5111100110

Advisor
Henning TiTi Ciptaningtyas, S.Kom., M.Kom.
Bagus Jati Santoso, S.Kom, Ph.D

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2017

(Halaman Ini Sengaja dikosongkan)

LEMBAR PENGESAHAN

Aplikasi Pendeteksi Serangan Pada HTTP Menggunakan n-gram

Tugas Akhir

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

Oleh:
Reinhard Ruben R.
NRP. 5111 100 110

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Henning TiTi Ciptaningtyas, S.Kom, M.Kom. ..... (pembimbing 1)
NIP: 19840708 201012 2 004

Bagus Jati Santoso, S.Kom, Ph.D ..... (pembimbing 2)
NIP: 510010004

SURABAYA
JULI , 2017

(Halaman Ini Sengaja dikosongkan)

Aplikasi Pendeteksi Serangan pada HTTP menggunakan n-gram

Nama Mahasiswa : Reinhard Ruben R
NRP : 5111 100 110
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I :Henning TiTi Ciptaningtyas, S.Kom., M.Kom.
Dosen Pembimbing II : Bagus Jati Santoso, S.Kom, Ph.D

Abstrak

Serangan digital saat ini jenisnya sangatlah banyak. Tiap hari jumlahnya juga selalu meningkat.Untuk mendeteksinya, ada banyak aplikasi yang menggunakan berbagai jenis metode untuk mendeteksi serangan – serangan yang ada. Riset – riset sebelumnya sudah menunjukkan bahwa analisis berlevel bite dari lalu lintas jaringan dapat digunakan untuk mendeteksi serangan dan analisis lalu lintas jaringan. Teknik ini tidak memerlukan pengetahuan khusus mengenai cara menjalankan aplikasi pada web-server atau memproses data yang masuk sebelumnya.

Pearson chi-squared test adalah salah satu algoritma yang digunakan untuk mendeteksi kemiripan antara 2 paket. Pearson chi-squared distance dapat diimplementasikan pada paket yang ada di dalam file. Paket yang masuk dianalisa dengan menggunakan algoritma N-gram untuk mendapatkan pola – pola. Paket ini kemudian akan dihitung jaraknya dengan menggunakan algoritma chi-squared distance dengan paket referensi yang sudah tersedia sebelumnya. Perhitungan ini kemudian akan diuji dengan hipotesa dari pearson chi-squared test untuk mengetahui apakah paket ini adalah merupakan paket serangan atau paket normal.

Uji coba dilakukan dengan melakukan analisa terhadap file – file yang berisikan paket bersih, dan file yang berisikan berbagai jenis serangan..Uji coba ini dilakukan dengan 4 jenis N-gram dan juga 5 jumlah data referensi yang berbeda. Hasil uji coba menunjukkan bahwa penggunaan metode 3-gram adalah

metode yang memiliki tingkat akurasi yang paling baik dengan tingkat false positive 3.4% dan tingkat akurasi pendeteksian serangan generic 64.077%, serangan shellcode 87.234%, dan serangan morphed-shellcode 79.404% dengan waktu analisa yang relatif cepat. Waktu yang dibutuhkan untuk melakukan proses perhitungan tingkat false positive adalah 1 jam 30 menit 36 detik dengan jumlah paket yang dianalisa 1000.

Kata kunci :, Analisis N-gram , Analisis traffic jaringan, Analisis Byte, Chi-squared Distance, Pearson chi-squared Test, Serangan HTTP

APPLICATION FOR HTTP ATTACK DETECTION USING N-GRAM

Student's Name	**: Reinhard Ruben R.**
Student's ID	**: 5111 100 110**
Department	**: Teknik Informatika FTIf-ITS**
Advisor I	**: Henning TiTi Ciptaningtyas, S.Kom., M.Kom.**
Advisor II	**: Bagus Jati Santoso, S.Kom, Ph.D**

ABSTRACT

Currently, there are many different kind of cyber attacks. This number keep growing everyday. To detect them, there are many kind of application that use numerous method to detect these attacks. Previous research has shown that byte-level analysis of network traffic can be useful for attack detection and network traffic analysis. This technique does not require any knowledge of application running on web-server or any pre-processing of incoming data.

Pearson chi-squared test is an algorithm used to detect the similiraty between 2 packets. This similiraity is calculated using pearson chi-squared distance. Pearson chi-squared distance will be implemented to the packet inside the file. The packet will first be analized with N-gram algorithm to get its pattern. This pattern will then be used to calculate the distance between the analized packet and the reference packet using the chi-squared distance algorithm. This calculation will then be used to test the hypothesis from the pearson chi-squared test to find out whether the analized packet is an attack packet or benign packet.

To test the accuracy of this method, the application will analyze both file filled with benign packet, and file filled with attack packet. This test is done using 5 kind of n-gram and 5 kind of reference data. The test result show us that 3-gram method is the best method between the 4 n-gram with the false positive rate of 3.4%

and 64.077% accuracy for generic attack detection, 87.234% for shellcode attack detection, and 79.404% for morphed-shellcode attack detection with a relatively fast running time. The time needed to analize 1000 benign paket with 3000 reference data is 1 hour 30 minute and 36 second.

Keywords : HTTP attacks , N-gram analysis , Network Traffic Analysis , Byte Analysis,Pearson Chi-squared Test

KATA PENGANTAR

Segala puji dan syukur kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Aplikasi Pendeteksi Serangan Pada HTTP menggunakan N-Gram".

Pengerjaan tugas akhir ini adalah momen bagi penulis untuk mengeluarkan seluruh kemampuan, hasrat, dan keinginan yang terpendam di dalam hati mulai dari masuk kuliah hingga lulus sekarang ini, lebih tepatnya di kampus Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak. Melalui lembar ini, penulis ingin secara khusus menyampaikan ucapan terima kasih kepada:

1. Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan baik
2. Ayah penulis, Sintong Hasibuan, dan Ibu penulis, Rosmalinda Hutapea, yang selalu memberikan dukungan, doa, perhatian, dan kasih sayang.
3. Adik penulis, Indri Asiani Larasati, dan kakak penulis, Naomi Marlini Novianti, yang telah membantu dan senantiasa memberi semangat untuk menyelesaikan studi penulis.
4. Ibu Henning Titi Ciptaningtyas selaku dosen pembimbing Tugas Akhir pertama dan yang telah memberikan arahan dalam pengerjaan Tugas Akhir ini.
5. Bapak Bagus Jati Santoso selaku dosen pembimbing Tugas Akhir kedua yang dengan sabar membimbing penulis dalam pengerjaan Tugas Akhir ini.
6. Bapak Radityo Anggoro selaku dosen koordinator Tugas Akhir yang telah membantu penulis atas segala sesuatu mengenai syarat-syarat dan terlaksananya sidang Tugas Akhir.

7. Bapak Darlis selaku Ketua Jurusan Teknik Informatika ITS yang selama ini memberikan bantuan kepada penulis.
8. Dosen-dosen Teknik Informatika yang dengan sabar mendidik dan memberikan pengalaman baru kepada penulis selama di Teknik Informatika.
9. Staf TU Teknik Informatika ITS yang senantiasa memudahkan segala urusan penulis di jurusan.
10. Rekan-rekan dan pengelola Laboratorium Komputasi Berbasis Jaringan yang telah memberikan fasilitas dan kesempatan melakukan riset atas Tugas Akhir yang dikerjakan penulis.
11. Rekan-rekan dan sahabat-sahabat penulis angkatan 2011 yang memberikan dorongan motivasi dan bantuan kepada penulis.
12. Pihak-pihak lain yang tidak sengaja terlewat dan tidak dapat penulis sebutkan satu per satu.

Penulis telah berusaha sebaik mungkin dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2017
Penulis

Reinhard Ruben R.

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR PSEUDOCODE

(Halaman ini sengaja dikosongkan)

DAFTAR KODE

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Bab ini memaparkan garis besar Tugas Akhir yang meliputi latar belakang, tujuan dan manfaat pembuatan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Kemajuan teknologi dewasa ini semakin pesat. Teknologi yang baru berkembang semakin meningkatkan produktivitas manusia. Banyak kantor – kantor yang sekarang memperbolehkan karyawannya untuk langsung bekerja dari rumah, dan hanya perlu mengisi daftar hadir pada web kantor tersebut. Banyak pula toko – toko yang membuat usahanya berbasis internet menggunakan aplikasi web saja.

Banyaknya penggunaan web ini tentu saja memudahkan manusia untuk melakukan kegiatannya. Kita dapat melakukan kegiatan kita tanpa harus beranjak dari kursi kita. Namun, dengan meningkatnya penggunaan web untuk melakukan kegiatan kita, hal ini juga membuat sebuah celah bagi penjahat – penjahat virtual untuk mencuri data pribadi kita atau merusak web yang digunakan orang lain. Hal ini dapat berakibat fatal bagi para pengguna web.

Menurut *Symantec internet security threat* [1] , sekitar 4500 jenis serangan terhadap web baru yang ditemukan setiap harinya. Server aplikasi web menjadi target yang umum untuk serangan – serangan tersebut karena server tersebut berkmounikasi dengan berbagai sistem lainnya.

Kebanyakan HTTP *traffic* hanya berisi karakter ASCII. Kita tidak menerima kode – kode yang bisa dijalankan pada paket HTTP yang kita terima. *Executable code* pada HTTP biasanya adalah sebuah penanda dari kemungkinan serangan injeksi *malware.* Distribusi byte dari ASCII jauh lebih terbatas dibandingkan dengan *executable code*, sehingga analasis dari distribusi byte pada paket HTTP bisa berguna untuk mendeteksi beberapa jenis serangan.

Aplikasi ini dibuat untuk mendeteksi serangan HTTP dengan menggunakan *n-gram analysis*, dan juga untuk menguji dan membuktikan akurasi dari analsis distribusi byte dalam mendeteksi serangan HTTP.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mendeteksi paket data yang berbahaya?
2. Bagaimana cara mengambil N-gram *sequence* dari paket yang datang?
3. Bagaimana cara menggunakan N-gram *sequence* untuk mendeteksi paket data yang berbahaya?
4. Metode N-gram mana yang paling baik untuk digunakan?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Serangan yang dideteksi adalah serangan pada HTTP.
2. Aplikasi yang dibangun adalah aplikasi berbasis desktop.
3. Aplikasi yang dibangun dengan menggunakan Microsoft Visual Studio.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini adalah untuk membuat sebuah aplikasi untuk mendeteksi serangan berbasis HTTP yang memiliki akurasi memuaskan, sehingga adaministrator server dapat mendeteksi serangan dengan lebih akurat dan mudah.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Membuktikan tingkat akurasi teknik n-gram untuk mendeteksi serangan berbasis HTTP.

2. Mempermudah administrator server dan web dalam mendeteksi serangan.

1.6 Metodologi

Pembuatan tugas akhir dilakukan menggunakan metodologi sebagai berikut:

A. Studi literatur

Tahap Studi Literatur merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literatur, diskusi, eksplorasi teknologi dan pustaka, serta pemahaman dasar teori yang digunakan pada topik tugas akhir. Literatur-literatur yang dimaksud disebutkan sebagai berikut:

1. Penggunaan library WinPcap
2. Algoritma Pearson chi square.

B. Perancangan perangkat lunak

Pada tahap ini dilakukan analisa awal dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang sedang dihadapi. Selanjutnya, dirumuskan rancangan sistem yang dapat memberi solusi terhadap permasalahan tersebut. Langkah yang akan digunakan pada tahap ini adalah sebagai berikut:

1. Pencarian dan pendataan materi yang akan digunakan untuk analisis serangan
2. Perancangan sistem dan mekanisme aplikasi untuk membaca dan menyimpan paket
3. Analisis algoritma dan formula dalam analisa paket.
4. Perancangan algoritma dan formula dalam analisa paket

C. Implementasi dan pembuatan sistem

Tahap implementasi merupakan tahap untuk membangun aplikasi beserta sistem yang terkait. Aplikasi ini akan dibangun dengan bahasa pemrograman C++ dengan Microsoft Visual C++.

D. Uji coba dan evaluasi
   Pada tahap ini akan dilakukan pengujian terhadap perangkat lunak menggunakan data atau skenario yang telah dipersiapkan sebelumnya yakni sebagai berikut:
   1. Pengujian blackbox
      Pengujian blackbox adalah pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. Pengujian ini dilakukan untuk menguji apakah proses kinerja aplikasi *game* ini sudah sesuai dengan kebutuhan pengguna atau tidak.
E. Penyusunan laporan tugas akhir
   Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan tugas akhir.

1.7 Sistematika Penulisan

Buku tugas akhri ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan tugas akhir, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II TINJAUAN PUSTAKA

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir ini.

BAB III ANALISIS DAN PERANCANGAN

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi

kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi pembangkitan area permainan, dan antarmuka permainan.

BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

(Halaman Ini Sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Teori-teori tersebut adalah Microsoft Visual Studio, C++, n-gram, K-Means Clustering dan Pearson chi-square test

2.1 Microsoft Visual Studio

Micorosft Visual Studio adalah sebuah *Integrated Develoment Environment(IDE)* dari Microsoft. Aplikasi ini digunakan untuk mengembangkan program komputer untuk *Microsoft Windows*, Situs Web, Apllikasi Web, Servis Web, dan Applikasi Perangkat Bergerak. Visual Studio menggunakan platform pengembangan software dari Microsoft seperti *Windows API , Windows Forms, Windows Presentation Foundation, Windows Store,* dan *Micorosoft Silverlight.*

Visual Studio memiliki fitur *Code Editor* yang mendukung *IntelliSense*. Debugger yang ada di dalam Visual Studio bisa dipakai sebagai debugger tingkat source dan juga debugger tingkat mesin. Fitur – fitur lainnya adalah seperti pendesain bentuk untuk membuat GUI aplikasi, pendesain *Web* , dan pendesain skema *database*. Visual studio menerima *plug-ins* yang menambahkan fungsional pada hampir semua level – termasuk menambahkan bantuan untuk sistem *source control* dan menambahkan *toolset* baru seperti penyunting dan pendesain virtual untuk bahasa khusus domain atau *toolset* untuk aspek lain dari *Software Development Lifecycle*.

Visual studio mendukung 36 bahasa pemrograman yang berbeda dan membolehkan *code editor* dan debugger untuk mendukung hampir semua bahasa pemrograman, jika servis untuk bahasa tersebut ada. Bahasa yang ada termasuk C , C++ dan C++/CLI , VB.NET , C# , F# , dan TypeScript. Dukungan untuk bahasa lain seprti Phyton , Ruby, Node.js, dan M dapat diperoleh dengan cara menginstall servis bahasa . Itu juga mendukung

XML/XSLT , HTML/XHTML, JavaScript dan CSS. Java dan J# didukung sebelum ini.

Microsoft menyediakan versi gratis dari Visual Studio dengan nama *Community Edition* yang mendukung plugins dan dapat diperoleh tanpa biaya. [2]

2.2 N-Gram

N-Gram adalah sebuah urutan dari sejumlah n barang dari sebuah rangkaian teks atau kata – kata. Rangkaian ini dapat berupa apa saja, misalnya huruf, kata – kata, atau kalimat sesuai dengan apa yang mau kita gunakan.

Sebuah N-gram berukuran 1 biasa disebut unigram, berukuran 2 biasa disebut bigram, berukuran 3 biasa disebut trigram. Ukuran yang lebih besar biasanya disebut sebagai *four-gram, five-gram*, dan seterusnya. [3]

Contoh penggunaan n-gram dapat dilihat pada tabel 2.1.

Tabel 2.1 Tabel N-gram

Kata/Kalimat	1-gram	2-gram	3-gram
Aasdasasda	A,a,s,d,a,a....	Aa,as,sd,da,as,....	Aas,asd,sda,das,....
Itu buku milik andi	Itu,buku,milik,andi	Itu buku, buku milik, milik andi	Itu buku milik, buku milik Andi

Metode N-gram ini digunakan pada tugas akhir ini karena metode ini adalah sebuah metode yang sering digunakan dalam *Text Mining* yang bisa digunakan sebagai model bahasa untuk meramalkan kata/huruf apa yang selanjutnya akan muncul.

Pada tugas akhir ini, metode N-gram digunakan untuk membuat pola – pola yang muncul pada paket yang akan digunakan untuk menghitung tingkat kemiripan antar paket.

2.3 *Pearson Chi-squared Test*

Pearson Chi-squared Test adalah sebuah tes statistik yang diterapkan pada suatu kumpulan data untuk menilai seberapa besar kemungkinan munculnya perbedaan yang bisa dilihat antara kumpulan data tersebut secara kebetulan. Tes ini cocok digunakan untuk suatu kumpulan data yang besar. Tes ini adalah tes yang paling banyak digunakan diantara *chi-squared Test(*Sebuah prosedur statistik yang hasilnya di nilai dengan membandingkannya dengan distribusi *chi-square)* yang lainnya . [4]

Tes ini menguji hipotesis *null* yang menyatakan bahwa distribusi frekuensi dari suatu kejadian yang dilihat dari contoh, mempunyai hasil yang konsisten dengan suatu teori distribusi lain.

Metode ini digunakan pada tugas akhir ini karena banyaknya jumlah data referensi dan data yang akan dianalisis. Metode ini juga digunakan karena dari antara metode *chi-squared Test* yang ada, metode ini adalah metode yang paling sering digunakan.

Pada tugas akhir ini, *Pearson chi-squared test* digunakan untuk membuktikan hipotess *null* yang sudah ditentukan sebelumnya.

2.4 Chi-squared Distance

Chi-squared Distance adalah sebuah metode untuk mencari jarak antara 2 histogram x=[x_1,x_2,....,x_n] dan y=[t_1,y_2,....,y_n]. Kedua histogram tersebut juga harus dinormalisasi dahulu, berarti isi dari kedua histogram tersebut harus berjumlah 1 [5].

Perhitungan jarak antar 2 histogram tersebut (D(x,y)) dihitung dengan menggunakan rumus yang biasa digunakan untuk mencari *chi-square*, dapat dilihat pada persamaan (2.1)

$$D(X,Y) = \sum_{i=1}^{n} \frac{(X_i - Y_i)^2}{Y_i} \quad (2.1)$$

Keterangan :

n = Jumlah data unik pada histogram
X_i = Nilai normalisasi dari nilai dari x_i
Y_i = Nilai normalisasi dari nilai dari y_i

Metode Chi-squared distance ini digunakan untuk menghitung tingkat kesamaan dari paket yang dianalisa dengan paket referensi yang sudah ditentukan sebelumnya.

2.5 WinPCap

WinPcap adalah sebuah aplikasi untuk menangkap dan mengirimkan paket-paket jaringan tanpa melalui *protocol stack* yang ada. Program ini juga mempunyai fitur – fitur lain seperti membuat statistic network dan mensupport paket capture jarak jauh.WinPcap adalah aplikasi gratis yang dikeluarkan dengan lisensi open source dari BSD. [6]

WinPcap terdiri atas sebuah *driver*, yang memperbolehkan sistem untuk mengakses jaringan tingkat rendah, dan sebuah *library* yang dapat digunakan untuk mengakses lapisan jaringan tingkat rendah. *Library* ini juga berisi versi *Windows* dari libpcap Unix API.

Library ini digunakan karena library ini adalah library yang dapat digunakan untuk menangkap dan menganalisis paket – paket yang dapat digunakan untuk *Windows*.

Pada tugas akhir ini, library ini digunakan untuk menangkap paket dan menyimpannya, dan juga membaca paket dari file yang sudah tersedia sebelumnya.

2.6 C++

Microsoft C++ adalah sebuah bahasa pemrograman multifungsi. Bahasa pemrograman ini memiliki fitur – fitur *programming* seperti *imperative , object – oriented , generic* , dan juga menyediakan fasilitas untuk manipulasi memori tingkat rendah. [7]

Bahasa pemrograman ini di desain dengan kecenderungan kepada sistem *programming* dan sistem besar yang memiliki resource terbatas, dengan performa, effisiensi, dan fleksibilitas kegunaanya sebagai pokok dari desain bahasa ini.

Bahasa pemrograman ini dibuat berbasiskan bahasa C dengan fitur- fitur baru yang dulunya tidak ada di C dan *library* standard yang lebih besar. Dulunya bahasa pemrograman ini dibuat sebagai tambahan dari bahasa pemrograman C , karena dibutuhkannya bahasa yang lebih efisien dan felksibel , dan juga menyediakan fitur – fitur tingkat tinggi yang dibutuhkan oleh organisasi – organisasi programmer.

(Halaman ini sengaja dikosongkan)

BAB III
ANALISIS DAN PERANCANGAN

Perancangan merupakan bagian penting dari pembuatan suatu perangkat lunak yang berupa perencanaan - perencanaan secara teknis aplikasi yang dibuat. Bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir ini. Berawal dari deskripsi umum aplikasi hingga perancangan proses, alur, dan implementasinya.

3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini dibangun aplikasi pendeteksi serangan HTTP yang memiliki kemampuan untuk memisahkan data bersih dan data berbahaya. Hasil akhir dari aplikasi ini adalah menyimpan data paket yang dicurigai yang nantinya bisa dianalisa lagi dengan aplikasi lain yang lebih canggih. Applikasi ini juga dapat melakukan penangkapan paket dari network yang kemudian akan disimpan untuk dianalisa di lain waktu.

Proses kerja aplikasi ini dimulai dengan membiarkan pengguna memilih ingin menangkap paket atau menganalisa paket. Jika pengguna ingin menangkap paket, maka aplikasi akan memberikan pilihan perangkat mana yang akan digunakan untuk menangkap paket, kemudian pengguna dapat memasukkan lokasi dan nama file tempat hasil tangkapan akan disimpan.. Jika pengguna memilih untuk menganalisa file, maka aplikasi akan membaca file yang ingin dianalisa. File ini berbentuk .Pcap. Kemudian aplikasi ini akan memanggil satu persatu paket dari file tersebut dan membandingkannya dengan dataset yang sudah tersedia. Aplikasi lalu mengambil pola - pola *n-gram* yang muncul pada paket yang kemudian akan di bandingkan dengan pola paket yang ada pada dataset. Jika paket yang dianalisa memliki kesamaan pola yang cukup dekat dengan paket yang ada pada dataset, maka paket tersebut dianggap paket bersih.

3.2 Perancangan Data

Percangan data adalah hal yang penting dalam aplikasi karena diperlukan data yang tepat agar aplikasi dapat berjalan dengan tingkat akurasi yang memuaskan. Aplikasi yang dibuat membutuhkan data referensi, data masukan dan data keluaran. Data referensi merupakan paket – paket bersih yang dipilih untuk menjadi pembanding apakah data masukan adalah paket berbahaya atau bukan. Data referensi ini berupa file berisikan byte dari paket – paket bersih. Data masukan berupa file hasil *sniffing* yang ingin dianalisa apakah paket – paket dalam file tersebut adalah serangan atau bukan. Data akhir adalah data lamanya aplikasi berjalan serta paket – paket mana saja yang dianggap sebagai paket serangan, setelah dilakukan pengecekan dengan data referensi.

- **Data Referensi**

Data Referensi yang digunakan adalah paket – paket yang diambil dari DARPA ‘99(1999) [8] dataset. Darpa ’99 dataset adalah dataset berisi simulasi serangan HTTP dari dan menuju pangkalan udara militer U.S. selama 5 hari. Dataset yang dipakai untuk data referensi diambil dari paket – paket yang ditangkap pada minggu pertama. Dataset minggu pertama ini adalah dataset berisikan paket – paket bersih, yang kemudian akan disimpan dan akan digunakan oleh aplikasi untuk membandingkan kesamaannya dengan data masukan untuk menentukan apakah data masukan adalah paket berbahaya atau bukan.

- **Data Masukan**

Data Masukan untuk paket – paket bersih, akan menggunakan paket – paket yang ditangkap pada minggu kedua dari DARPA’99(1999) dataset.

Untuk data paket serangan digunakan dataset serangan yang diperoleh dari Perdisci [9]. Jenis – jenis serangan yang didapat dari Perdisci adalah :

- Serangan *Generic* : ini adalah sebuat dataset yang dapat diakses oleh publik yang disediakan oleh penulis *Ingham and Inoue(2007)* [10]. Dataset ini berisi 66 jenis serangan seperti : *Denial of Service, URL decoding error attacks, shellcode attacks,* dan serangan – serangan lain yang menyebabkan kebocoran informasi. Menurut *Ariu et al* [11] , paket – paket yang terdapat pada dataset ini memiliki kemiripan dengan paket – paket bersih, sehingga paket – paket ini mungkin akan sulit dideteksi.
- Serangan *Shellcode :* serangan *shellcode* adalah serangan yang mencoba untuk memasukkan kode dengan mempergunakan vulnerabilitas *buffer overflow* pada mesin yang di target. Serangan – serangan ini umumnya mudah dideteksi karena mereka memiliki byte – byte yang tidak dimiliki oleh paket bersih.
- *CLET attacks : Clet* adalah sebuah pembuat *polymorphic worm* yang membuat beberapa versi terenkripsi dari malware(CLET) [12]. Dataset ini dibuat dengan menerapkan CLET pada serangan Shellcode.

- **Data Keluaran**

Data keluaran dari aplikasi ini adalah lamanya aplikasi ini berjalan, dan paket – paket mana saja yang dianggap sebagai paket serangan setelah dibandingkan dengan data referensi.

3.3 Perancangan Algoritma

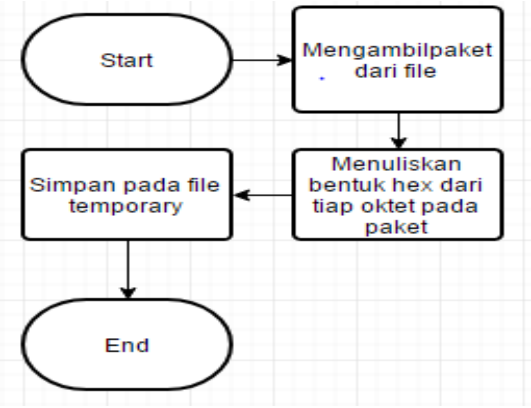
Perancangan algoritma merupakan tahap untuk membentuk alur dalam penerapan algoritma *n-gram* untuk mendapatkan pola – pola yang muncul pada paket, algoritma *Chi-squared distance* untuk mendapatkan tingkat kemiripan antara paket yang dianalisa dan paket referensi, dan *Pearson Chi-squared test* untuk mendapatkan hipotesis akhir.

3.3.1 Desain Secara Umum

Pembuatan dataset referensi dimulai dengan mengambil paket dari DARPA'99 dataset. Paket yang diambil adalah paket bersih yang diambil bentuk hexadecimalnya dan disimpan dalam text file yang akan digunakan sebagai data referensi. Setelah data referensi terbentuk, maka aplikasi dapat memulai proses analisis paket. Pertama, aplikasi akan mengambil sebuah paket yang ingin dianalisa dari file yang disediakan oleh pengguna. Kemudian, aplikasi akan mengambil nilai – nilai hexadecimal dari tiap byte yang terdapat pada paket tersebut. Setelah itu, aplikasi akan mencari pola – pola n-gram yang terdapat pada hexadecimal tersebut dan menghitung frekuensinya. Setelah itu, paket tersebut akan dihitung kemiripinnya dengan data referensi dengan menggunakan algoritma *Chi-squared distance.* Setelah nilai kemiripinnya didapat, kemudian akan diuji dengan *pearson chi-squared test* untuk mennentukan apakah paket tersebut berbahaya atau tidak.

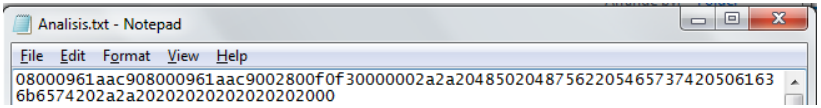
3.3.2 Tahap Pra-Proses Analisa Data

Pada tahap ini, dilakukan pengambilan dan pemrosesan paket dari file yang ingin dianalisa. Data masukan pada tahap ini adalah lokasi dari paket yang ingin dianalisa. Diagram alur dari tahap ini dapat dilihat pada Gambar **3.1**.

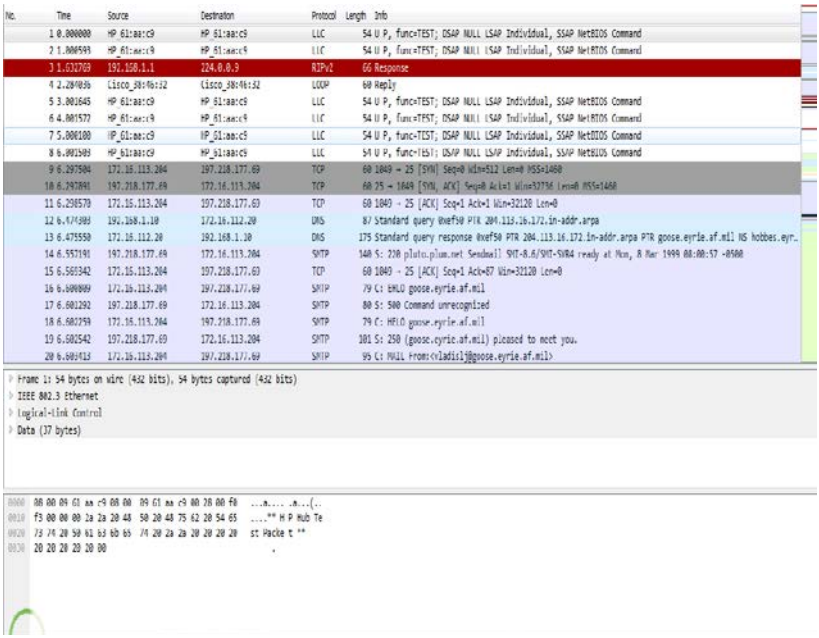


Gambar 3.1 Diagram Alur tahap pra-proses data

Pertama, program akan mengambil sebuah paket dari file yang akan dianalisa. Format dari file ini biasanya .pcap atau .tcpdump. Paket – paket dalam file ini masih berbentuk byte, sehingga untuk membacanya secara langsung sangat sulit. Untuk membaca paket – paket ini biasanya menggunakan applikasi wireshark atau semacamnya. Tampilan paket – paket dalam file pada aplikasi dapat dilihat pada Gambar 3.3 . Pada bagian bawah dapat dilihat kumpulan angka – angka. Angka – angka itu adalah byte – byte yang akan kita gunakan untuk melakukan klasifikasi serangan atau bukan. Program akan mengambil angka – angka tersebut, dan kemudian menyimpannya ke dalam sebuah file temporary. Bentuk paket dalam file temporary dapat dilihat pada Gambar 3.2.



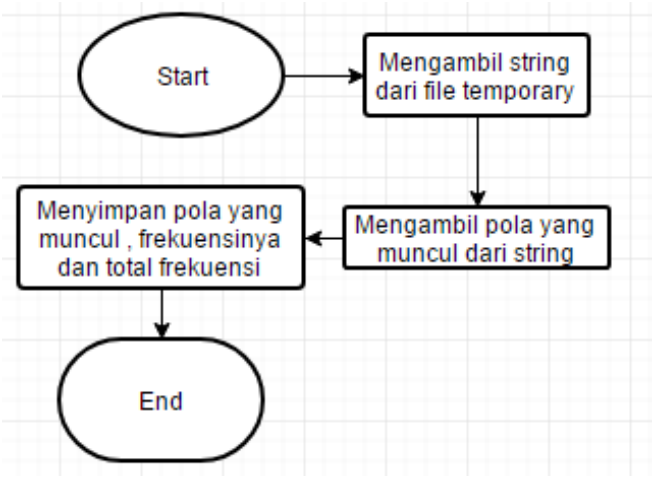
Gambar 3.2 Bentuk Paket Dalam File Temporary



Gambar 3.3 Bentuk Paket Dalam Wireshark

3.3.3 Tahap Pembentukan Pola N-gram

Di tahap ini dilakukan pengambilan pola n-gram dari paket yang ingin dianalisa. Pertama, aplikasi akan memanggil semua data yang ada pada file temporary. Data yang ada pada file temporary ini adalah data berbentuk hexadecimal. Kemudian aplikasi akan mengambil *n-gram* dari hexadecimal tersebut dan menyimpan pola – pola yang muncul, frekuensinya dan total frekuensi dari semua pola yang muncul. Hasil frekuensinya tersebut kemudian akan dinormalisasi agar dapat digunakan untuk proses perhitungan jaraknya. Alur pembentukan pola *n-gram* dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram Alur tahap pembentukan pola n-gram

Disini, program akan membentuk pola – pola n-gram yang muncul pada paket yang dianalisa. Program akan membentuk pola – pola n-gram dari data yang ada dalam file temporary tadi. Pola – pola n-gram yang muncul pada paket dapat dilihat pada Tabel 3.1. Untuk contoh ini, digunakan metode 2-gram.

Tabel 3.1 Pola – Pola *N-gram* pada paket

Pola	Frekuensi
08	2
80	3
00	14
09	2
96	2
61	3
1a	2
Aa	2
Ac	2

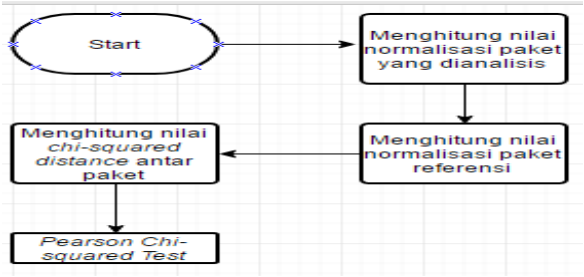
Pola	Frekuensi
85	1
50	2
87	1
76	1
65	1
62	1
22	1
05	2
54	1

C9	2
90	2
02	13
28	1
0f	2
F0	1
F3	1
30	1
2a	4
A2	4
20	15
04	2
48	2

46	1
65	2
57	2
73	1
37	1
74	2
42	2
06	1
16	1
63	1
36	1
6b	1
B6	1

3.3.4 Tahap Penghitungan Nilai *Chi-square Distance*

Setelah didapatkan pola – pola yang muncul pada paket dan frekuensinya, program akan menghitung nilai normalisasi dari masing – masing pola yang terdapat pada paket yang ingin dianalisa, aplikasi kemudian akan menghitung nilai normalisasi dari masing – masing pola yang terdapat pada paket referensi. Kemudian, aplikasi akan menghitung *chi-squared distance* antara 2 paket tersebut. Alur penghitungan nilai *chi-square distance* dapat dilihat pada Gambar 3.5.



Gambar 3.5 Tahap Penghitungan Nilai *Chi-Square Distance*

Disini, program akan menghitung nilai jarak antara paket yang kita analisis tadi dengan paket referensi yang sudah tersedia sebelumnya. Pertama – tama, program akan menghitung nilai normal dari setiap pola yang ada pada paket yang akan dianalisa. Nilai normalisasi dari paket yang dianalisa dapat dilihat pada Tabel 3.2

Tabel 3.2 Nilai normalisasi Paket Analisa

Pola	Normalisasi
08	0.018181818
80	0.027272727
00	0.127272727
09	0.018181818
96	0.018181818
61	0.027272727
1a	0.018181818
aa	0.018181818
ac	0.018181818
c9	0.018181818
90	0.018181818
02	0.118181818
28	0.009090909
0f	0.018181818
f0	0.009090909
f3	0.009090909
30	0.009090909
2a	0.036363636
a2	0.036363636
20	0.136363636
04	0.018181818
48	0.018181818

Pola	Frekuensi
85	0.009090909
50	0.018181818
87	0.009090909
76	0.009090909
65	0.009090909
62	0.009090909
22	0.009090909
05	0.018181818
54	0.009090909
46	0.009090909
65	0.018181818
57	0.018181818
73	0.009090909
37	0.009090909
74	0.018181818
42	0.018181818
06	0.009090909
16	0.009090909
63	0.009090909
36	0.009090909
6b	0.009090909
b6	0.009090909

Nilai – nilai pada Tabel 3.2 didapat dengan membagikan frekuensi pola dengan total frekuensi semua pola yang ada pada paket. Setelah nilai normalisasi paket didapat, selanjutnya program akan mencari pola – pola yang muncul pada paket referensi dan nilai normalnya. Bentuk paket – paket referensi dapat dilihat pada **Error! Reference source not found.**Gambar 3.6 , dan nilai normalnya pada Tabel 3.3 .

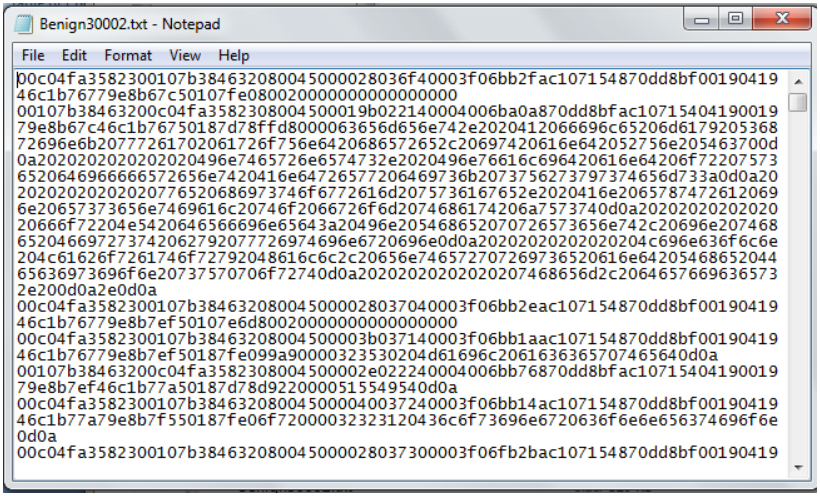
Tabel 3.3 Nilai Normalisasi Paket Referensi

Pola	Normalisasi
00	0.130841
0c	0.009346
c0	0.009346
04	0.028037
4f	0.009346
fa	0.018692
a3	0.009346
35	0.009346
58	0.009346
82	0.009346
23	0.009346
30	0.009346
01	0.028037
10	0.028037
07	0.028037
7b	0.009346
b3	0.009346
38	0.009346
84	0.009346
46	0.018692
63	0.009346
32	0.009346
20	0.018692

Pola	Nomalisasi
6b	0.009346
bb	0.009346
b2	0.009346
2f	0.009346
ac	0.009346
c1	0.018692
71	0.009346
15	0.009346
54	0.009346
48	0.009346
87	0.009346
70	0.009346
0d	0.009346
dd	0.009346
d8	0.009346
8b	0.018692
bf	0.009346
19	0.018692
90	0.009346
41	0.009346
94	0.009346
6c	0.009346
1b	0.009346

08	0.018692
80	0.028037
45	0.009346
50	0.018692
02	0.018692
28	0.009346
03	0.018692
36	0.009346
6f	0.009346
f4	0.009346
40	0.009346
3f	0.009346
f0	0.018692
06	0.009346

b7	0.009346
76	0.009346
67	0.009346
77	0.009346
79	0.009346
9e	0.009346
e8	0.009346
b6	0.009346
7c	0.009346
c5	0.009346
7f	0.009346
fe	0.009346
e0	0.009346



Gambar 3.6 Bentuk Paket Referensi

Dapat dilihat dari Tabel 3.3 dan Tabel 3.2, bahwa pola – pola yang muncul pada paket tidak semuanya sama. Maka, dalam

penghitungan nilai *chi-squared distance*-nya, jenis pola yang akan digunakan adalah pola – pola yang muncul pada paket referensi. Jadi, jika sebuah pola ada pada paket referensi, namun tidak ada pada paket yang dianalisa, maka frekuensi pola tersebut pada paket yang dianalisa dianggap 0. Dengan menggunakan metode *chi-squared distance* maka program akan menghitung jarak antara paket referensi dan paket yang dianalisa.

$$\mathrm{D}^2 = \frac{(0.127272727 - 0.130841)^2}{0.130841} + \frac{(0 - 0.009346)^2}{0.009346} + \frac{(0 - 0.009346)^2}{0.009346} + \ldots + \frac{(0 - 0.009346)^2}{0.009346} = 0.6$$

Nilai ini akan kemudian akan digunakan untuk menguji hipotesa yang sudah ditentukan sebelumnya.

3.3.5 Tahap Analisis *Pearson Chi-square Test*

Untuk melakukan analisis *Pearson Chi-square Test*, pertama akan dibentuk dulu hipotesis sebagai berikut :

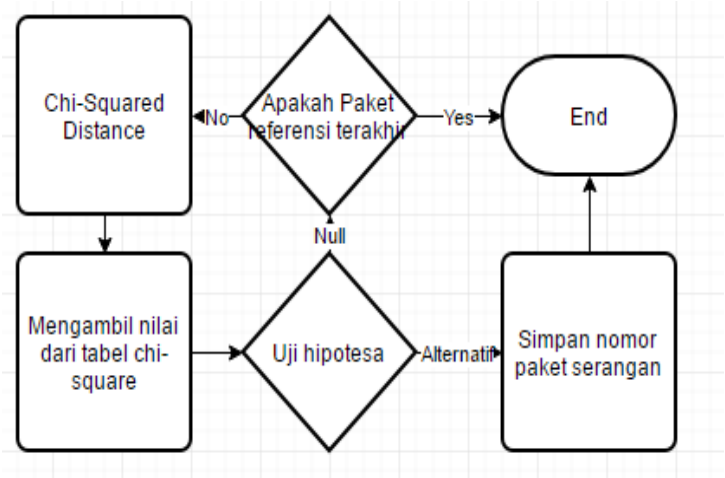
$$H_0 : D^2 \leq X^2(\alpha, b-1)$$
$$H_1 : D^2 > X^2(\alpha, b-1)$$

Keterangan dari hipotesis ini adalah :

1. Hipotesa *Null* disini berarti paket adalah serangan, dan hipotesa alternatifnya adalah paket bukanlah serangan.
2. D^2 adalah *chi-square distance* antara 2 paket.
3. X^2 adalah nilai dari chi-square table dengan nilai α = 0.05 dan *degree of freedom* b-1.
4. α yang digunakan pada tugas akhir ini adalah $\alpha = 0.05$.

5. *b* adalah jumlah pola unik yang muncul pada paket referensi.

Analisis ini Dilakukan untuk mengetahui tingkat kemiripan dari paket yang akan dianalisa dan paket referensi. Jika paket yang dianalisa memliki kemiripan diatas batas, maka hipotesa alternatif berhasil, yang berarti paket tersebut bukanlah paket berbahaya. Jika paket yang dianalisa memiliki tingkat kemiripan dibawah dari batas, maka hipotesis nullnya berhasil, yang berarti paket tersebut adalah paket berbahaya. Diagram alur analisis ini dapat dilihat pada Gambar 3.7.



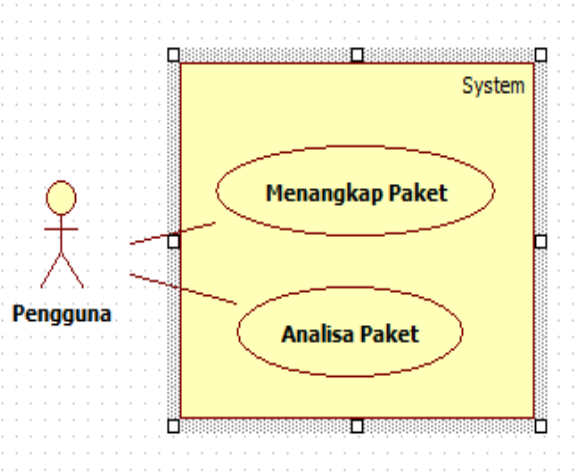
Gambar 3.7 Diagram Alur Tahap Pengklasifikasian Paket

Pada tahap ini, program akan melakukan perbandingan antara nilai *chi-squared distance* antara paket yang dianalisa dan paket referensi dengan nilai dari table *chi-squared* dengan nilai α = 0.05 , dan *degree of freedom* b-1. Dari perhitungan *chi-squared distance* tadi, kita mendapatkan nilai 0.6. Nilai dari $x^2(0.05,72)$ adalah 92.80827009. Karena nilai dari *chi-squared distance* lebih

kecil dari nilai x^2, berarti paket adalah serangan. Setelah itu , program akan mencek apakah ini adalah data referensi terakhir atau bukan. Jika bukan, program akan melakukan perhitungan *chi-squared distance* dengan data referensi selanjutnya. Namun, jika ini adalah data referensi terakhir, program akan menandakan paket ini bukanlah paket yang berbahaya dan akan melanjutkan analisa ke paket berikutnya.

3.4 Use Case Sistem

Use case sistem merupakan diagram kebutuhan yang menggambarkan fungsionalitas sistem dan aktor – aktornya. Berdasarkan pada Gambar 3.8 hanya ada 1 aktor yang terlibat yaitu pengguna. Pengguna berinteraksi dengan aplikasi untuk menjalankan segala fungsionalitas aplikasi. *Use case* pada aplikasi secara umum dijelaskan pada Tabel 3.4 .



Gambar 3.8 Use Case Sistem

Tabel 3.4 Tabel Deskripsi Use Case Sistem

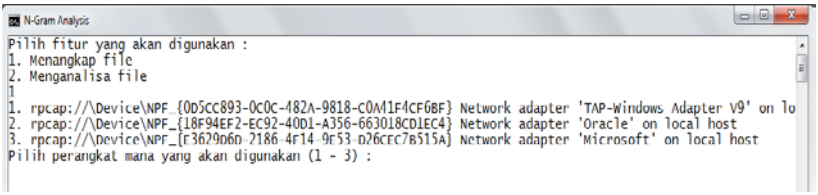
No	Kode	Nama	Keterangan
1.	UC-01	Melakukan Penangkapan Paket	Pengguna dapat menangkap paket – paket yang masuk dan dikirim oleh komputer pengguna
1	UC-02	Melakukan Analisa File	Pengguna dapat melakukan analisis file yang diinginkan menggunakan n-gram yang dipilih pengguna.

3.5 Perancangan Antarmuka Sistem

Pada Tugas Akhir ini, antarmuka sistem hanyalah berupa command console saja. Antarmuka Analisa File adalah antarmuka yang ditampilkan pada saat pengguna ingin melakukan analisa file hasil sniffing.

3.5.1 Perancangan Antarmuka Penangkapan Paket

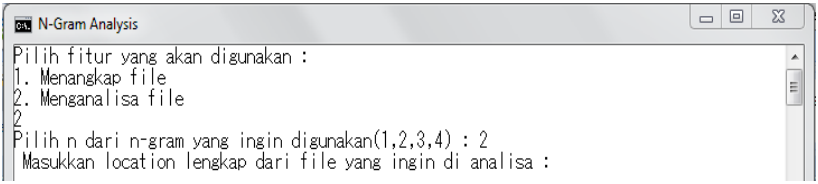
Perancangan antarmuka penangkapan paket ini meliputi beberapa komponen. Ketika aplikasi dibuka, aplikasi akan meminta pengguna untuk memilih menangkap atau menganalisa paket. Jika pengguna memilih untuk melakukan penangkapan paket, aplikasi akan menunjukkan perangkat mana saja yang ada di computer pengguna. Setelah pengguna memilih perangkat yang akan digunakan, maka aplikasi akan melakukan penangkapan paket. Antarmukanya dapat dilihat pada Gambar 3.9.



Gambar 3.9 Gambar Antarmuka Penangkapan Paket

3.5.2 Perancangan Antarmuka Analisa File

Perancangan antarmuka analisa file ini meliputi beberapa komponen. Ketika aplikasi dibuka , aplikasi akan meminta pengguna untuk memasukkan jumlah n pada n-gram yang ingin digunakan untuk analisa(1-4). Kemudian pengguna akan diminta untuk memasukkan lokasi file yang ingin dianalisa. Antarmukanya dapat dilihat pada Gambar 3.10 .



Gambar 3.10 Gambar Antarmuka Analisa File

BAB IV
IMPLEMENTASI

Pada Bab ini dibahas mengenai implementasi dari perancangan perangkat lunak. Di dalamnya mencakup proses penerapan dan pengimplementasian algoritma, dan antar muka yang mengacu pada rancangan yang telah dibahas sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi dari tugas akhir dijelaskan pada Tabel **4.1**.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat Keras	Prosesor : Intel® Core™ i5-2430M CPU @2.40 GHz (4CPUs), ~2.4GHz Memori : 4GB
Perangkat Lunak	Sistem Operasi : Windows 7 Home Premium 64-bit Perangkat Pengembang : Microsoft Visual Studio 2012 Direct X Version : DirectX11

4.2 Implementasi Applikasi

Implementasi dari masing – masing fungsi utama dituliskan menggunakan *pseudocode* berdasarkan metode - metode utama yang ada pada applikasi. Penjelasan implementasi yang akan dijelaskan adalah pembuatan n-gram, penghitungan frekuensi pola, mengganti nilai hex menjadi decimal, menentukan pola yang muncul, menghitung nilai normal paket, menghitung jarak antar paket dan penentuan serangan.

4.2.1 Implementasi Pembuatan N-gram

Implementasi ini berfungsi untuk mendapatkan pola – pola n-gram yang muncul pada paket. Pada fungsi ini, pola yang muncul tidak disimpan karena tempat penyimpanan pola yang muncul ada pada fungsi lain. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.1** dan Kode **4.1**.

```
Algoritma n-gram
Input : nilai n dari n-gram
        String data yang mau diambil n-gramnya
        Panjang string yang mau diambil n-gramnya
Output : pola sebagai n-gram yang muncul
For each n character from string do
    pola = n
```

Pseudocode 4.1 Pseudocode Pembuatan N-gram

```
//Mencari n-gram dari string
// b = n dari n-gram;input = string berisikan nilai – nilai hex pada paket ;e = variable untuk n-gram
Int b,c;
String input;
c = input.length();

for(int d=0 ; d < (c-b) ; d++)
{
    e = input.substr(d,b);
}
```

Kode 4.1 Kode Pembuatan N-gram

4.2.2 Implementasi Penghitungan frekuensi Pola

Implementasi ini berfungsi untuk mendapatkan frekuensi dari pola – pola yang muncul pada paket. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.2** dan Kode **4.2**.

```
Algoritma Pola
Input : e sebagai pola n-gram
Output : frekuensi dari pola e

For each e from string do
    frekuensi e ++
```

Pseudocode 4.2 Pseudocode Perhitungan Frekuensi Pola

```
//Mencari Pola dari n-gram string
// b = n of n-gram ; input = string of hex of packet ; e = container of n-gram ; Frekuensi = frekuensi dari sebuah pola ;
Int b,c;
String input;
Vector<int> frekuensi(pow(16,b-1)];
c = input.length();

for(int d=0 ; d < (c-b) ; d++)
{
     e = input.substr(d,b);
     frekuensi[e]++;
}
```

Kode 4.2 Kode Perhitungan Frekuensi Pola

4.2.3 Implementasi Mengganti Hex Menjadi Desimal

Implementasi ini berfungsi untuk merubah nilai hex menjadi nilai desimalnya agar lebih mudah diproses. Fungsi ini dipakai untuk memudahkan penyimpanan dan pemanggilan pola – pola yang muncul. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.3** dan Kode **4.3**.

```
Algoritma hextodec
Input : string hex
Output : string desimal

For each hex in string hex do
    e=0;
    If hex between 48 and 57
        Temp + (hex-48) * 16^(panjang hex-e-1)
    Else If hex between 65 and 70
        Temp + (hex -55) * 16^(panjang hex-e-1)
    Else if hex between 97 and 102
        Temp + (hex – 87) * 16^(panjang hex-e-1)
    e++
```

Pseudocode 4.3 Pseudocode Mengganti Hex menjadi Desimal

```
//Mengganti hex menjadi desimal
Int hextodec(string hex)
{
    int temp=0,a=0;
    int length;
    length = hex.length();

    for(int e=0; e<length;)
        {

            if (hex[e] >= 48 && hex [e] <= 57)
            {
                temp += (((int)(hex[e])) - 48) * pow(16, length - e
- 1);
            }

            else if ((hexa[e] >= 65 && hexa[e] <= 70))
            {
                  temp += (((int)(hex[e])) - 55) * pow(16, length - e - 1);
              }

            else if (hex [e] >= 97 && hex [e] <= 102)
```

```
                {
                    temp += (((int)(hexr[e])) - 87) * pow(16, length - e - 1);
                }

            e++;

        }

    return temp;
}
```

Kode 4.3 Kode Mengganti Hex menjadi Desimal

4.2.4 Implementasi Menentukan Pola Muncul

Implementasi ini berfungsi untuk menentukan pola – pola mana saja yang muncul dari string paket, sehingga pada perhitungan – perhitungan selanjutnya tidak perlu dipanggil semua pola yang muncul hanya pola – pola yang muncul pada paket referensi saja.. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.4** dan Kode **4.4**

```
Algoritma Pola Muncul
Input : e sebagai pola dari n-gram
        b sebagai jumlah n dari n-gram
        c sebagai panjang dari string
output : flag berisi pola yang muncul
for each e do
    if frekuensi(e) = 0
        flag(e) = 1
    frekuensi(e) + 1
```

Pseudocode 4.4 Pseudocode Pola Muncul

```
//Menentukan Pola Muncul
// b = n of n-gram ; input = string of hex of packet ; e = container of n-gram ; Frekuensi = frekuensi dari sebuah pola ;
```

```
Int b,c,totalfrekuensi;
String input;
Vector<int> frekuensi(pow(16,b));
Vector<int> flag(pow(16,b));

c = input.length();
f=0;

for(int d=0 ; d < (c-b) ; d++)
{
      e = input.substr(d,b);

      if(!frekuensi[hextodec(e)])
      {
            flag[f] = e;
            f++;
      }

      frekuensi[e]++;
     totalfrekuensi++;
}
```

Kode 4.4 Kode Menentukan Pola Muncul

4.2.5 Implementasi Menghitung Nilai Normal

Implementasi ini berfungsi untuk menentukan Nilai Normal dari masing – masing pola yang muncul pada paket. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.5** dan Kode **4.5**.

```
Algoritma menghitung nilai normal
Input : frekuensi sebagai nilai frekuensi pola
        totfrek sebagai total frekuensi dari pola
output : nilai normalisasi pola

normalisasi = frekuensi/totfrek
```

Pseudocode 4.5 Pseudocode Nilai Normal

```
//Menghitung nilai normalisasi
Double normalisasi(int frekuensi, int totalfrekuensi)
{
    Double normal;
    Normal = (double)frekuensi/totalfrekuensi;
    return normal
}
```

Kode 4.5 Kode Nilai Normal

4.2.6 Implementasi Perhitungan Kesamaan Antar Paket

Implementasi ini berfungsi untuk menentukan tingkat kesamaan antara paket yang dianalisa dengan paket referensi. Nilai dari fungsi ini kemudian akan digunakan untuk menentukan apakah paket ini berbahaya atau tidak. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.6** dan Kode **4.6**.

```
Algoritma kesamaan
Input : normal sebagai nilai normal dari pola pada string yang akan dianalisa
        normalA sebagai nilai normal dari pola pada string referensi
       flag sebagai pola yang muncul pada string referensi
output : Jarak antara kedua string

for each flag do
    kalkulasi + (normal(flag) – normalA(flag))^2 / normalA(flag)
```

Pseudocode 4.6 Pseudocode Kesamaan

```
//Perhitungan Kesamaan Antar Paket
//normal = nilai normalisasi pola dari paket yang akan dianalisa
//normalA = nilai normalisasi pola dari paket referensi
//p = jumlah pola unik dari paket referensi
//flag = pola yang muncul pada paket referensi
double normalisasi,normalisasiA,kalkulasi;

for(int d=0; d<p; d++)
```

```
{
    kalkulasi += pow(normal[flag[d]] – normalA[flag[d]]) / normalA[flag[d]];
{
```

Kode 4.6 Kode Kesamaan

4.2.7 Implementasi Penentuan Serangan

Implementasi ini berfungsi untuk menentukan apakah paket yang di analisa merupakan paket berbahaya atau bukan. Disini, nilai dari fungsi kesamaan antar paket akan dibandingkan dengan nilai chisqrnya untuk menentukan apaka paket tersebut merupakan paket berbahaya atau bukan. Kode untuk implementasi ini dapat dilihat pada Pseudocode **4.7** dan Kode **4.7**.

```
Algoritma Penentuan
Input : jarak sebagai jarak antara 2 string
        Chisqr sebagai nilai dari tabel chisqr
Output : 1 jika paket adalah serangan dan 0 jika paket bukan serangan

If jarak > chisqr
    return 1
else
    return 0
```

Pseudocode 4.7 Pseudocode Penentuan Serangan

```
//Menentukan Paket Serangan atau bukan.
//chisqr = nilai chisqr dengan b-1 degree of freedom dan α = 0.05.
double kalkulasi;

If(kalkulasi > chisqr)
{
    return 1;
}
```

Kode 4.7 Kode Penentuan Serangan

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan menggunakan metode kotak hitam berdasarkan skenario yang telah ditentukan dan pengujian dilakukan dengan menjalankan langsung apllikasi.

5.1 Lingkungan Uji Coba

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini dapat dilihat pada Tabel **5.1**

Tabel 5.1 Tabel Lingkungan Uji Coba

Perangkat Keras	Prosesor : Intel® Core™ i5-2430M CPU @2.40 GHz (4CPUs), ~2.4GHz Memori : 4GB
Perangkat Lunak	Sistem Operasi : Windows 7 Home Premium 64-bit Perangkat Pengembang : Microsoft Visual Studio 2012 Direct X Version : DirectX11

5.2 Pengujian Performa Aplikasi

Pengujian dilakukan untuk mengetahui performa dari algoritma yang telah dibuat pada tugas akhir ini. Pengujian yang dilakukan adalah pengujian *running time, CPU usage, dan memory usage.*

5.2.1 Skenario Pengujian Performa

Skenario pengujian performa aplikasi ini dapat dilihat pada Tabel **5.2**

Tabel 5.2 Tabel Pengujian Performa Aplikasi

Deskripsi	Bertujuan untuk mengetahui performa aplikasi dalam menganalisa paket
Langkah Pengujian	1. Pilih jumlah data referensi yang ingin diuji. 2. Pilih sebuah file yang berisi paket – paket baik. 3. Jalankan program dan pilih jumlah n-gram yang ingin diuji. 4. Biarkan program berjalan hingga 1000 paket sudah dianalisa 5. Catat waktu yang digunakan, penggunaan CPU, dan penggunaan memori aplikasi.

5.2.2 Hasil Pengujian Performa

Hasil Pengujian perfroma dikhususkan untuk mengetahui jumlah waktu yang dibutuhkan aplikasi untuk menanalisa paket, jumlah CPU yang digunakan selama aplikasi berjalan, dan jumlah memori yang dipakai aplikasi. Pengujian dilakukan untuk setiap n-gram pada berbagai jumlah data referensi.

Tabel 5.3 Tabel Hasil Pengujian Running Time

N-gram	**Jumlah Data Referensi**				
	1.000	**2.000**	**3.000**	**4.000**	**5.000**
1-gram	1003 detik	2206 detik	3443 detik	4271 detik	5171 detik

2-gram	1071 detik	2098 detik	3517 detik	4661 detik	5365 detik
3-gram	1311 detik	2684 detik	4092 detik	5436 detik	6650 detik
4-gram	2263 detik	4699 detik	6386 detik	8921 detik	10929 detik

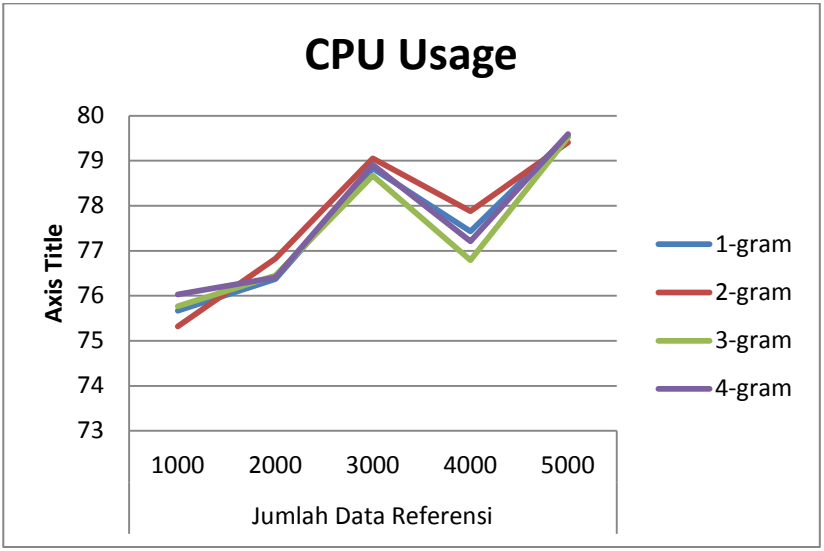
Gambar 5.1 Grafik Running Time

Pada Tabel **5.3** dan Gambar **5.1** dapat dilihat bahwa semakin banyak n-gram yang digunakan, waktu yang dibutuhkan untuk

melakukan analisa juga semakin banyak. Dapat dilihat bahwa antara 3-gram dan 4-gram ada perbedaan waktu yang signifikan dibandingkan antara 3-gram dan 2-gram. Selain dari jumlah n-gram yang dipakai, jumlah data referensi yang digunakan juga mempengaruh kecepatan analisa paket.

Tabel 5.4 Tabel Penggunaan CPU

n-gram	Jumlah Data Referensi				
	1.000	2.000	3.000	4.000	5.000
1-gram	75.67	76.37	76.33	77.43	77.13
2-gram	75.32	76.83	76.51	77.88	76.93
3-gram	75.77	76.45	75.97	76.79	76.85
4-gram	76.03	76.41	76.24	77.21	77.02

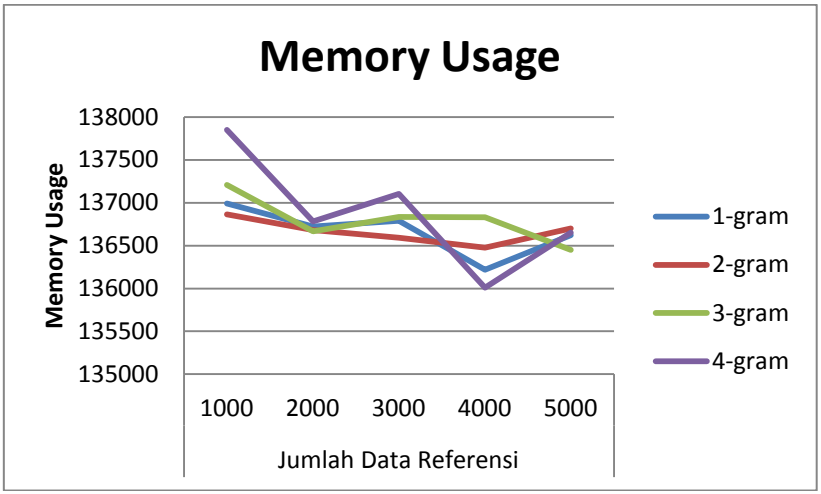


Gambar 5.2 Grafik Penggunaan CPU

Pada Tabel **5.4** dan Gambar **5.2** dapat dilihat bahwa jumlah data referensi dan n-gram yang digunakan tidak terlalu berpengaruh terhadap jumlah CPU yang dipakai oleh aplikasi.

Tabel 5.5 Tabel Penggunaan Memori

n-gram	**Jumlah Data Referensi**				
	1.000	2.000	3.000	4.000	5.000
1-gram	136.992 kb	136.721 kb	136.788 kb	136.217 kb	136.623 kb
2-gram	136.863 kb	136.682 kb	136.593 kb	136.475 kb	136.701 kb
3-gram	137.208 kb	136.665 kb	136.836 kb	136.832 kb	136.451 kb
4-gram	137.851 kb	136.783 kb	137.103 kb	136.011 kb	136.652 kb



Gambar 5.3 Grafik Penggunaan Memori

Pada Tabel **5.5** dan Gambar **5.3** dapat dilihat bahwa memori yang digunakan aplikasi tidak terpengaruh banyak dari jumlah data referensi dan n-gram yang digunakan.

5.3 Pengujian Akurasi

Pengujian akurasi ini digunakan untuk mengetahui akurasi aplikasi ini dalam membedakan paket serangan dan paket bersih. Pengujian yang dilakukan adalah pengujian *false positive* dan pengujian *true positive*. Untuk pengujian akurasi ini, akan digunakan 3000 data referensi.

5.3.1 Skenario Pengujian False Positive

Skenario pengujian false positive aplikasi ini dapat dilihat pada Tabel **5.6**

Tabel 5.6 Tabel Skenario Pengujian False Positive

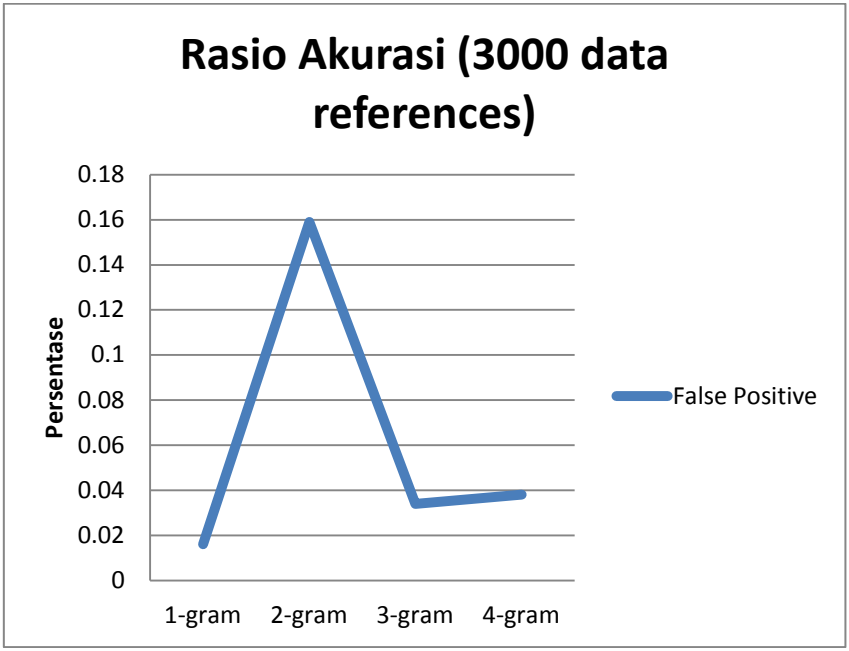
Deskripsi	Bertujuan untuk mengetahui rasio *false positive* dari aplikasi ini
Langkah Pengujian	1. Pilih file yang berisi paket – paket bersih 2. Jalankan program dan pilih n-gram yang dipakai untuk pengujian 3. Biarkan program berjalan hingga 1000 paket sudah dianalisa 4. Catat ratio *false positive*-nya

5.3.2 Hasil Pengujian False Positive

Hasil pengujian ini adalah untuk mengetahui n-gram mana yang paling bagus akurasinya dalam memisahkan paket bersih.

Tabel 5.7 Tabel Hasil Pengujian False Positive (Presisi 3 digit)

n-gram	False Positive
1-gram	0.016
2-gram	0.159
3-gram	0.034
4-gram	0.038



Gambar 5.4 Grafik Pengujian False Positive

Pada Tabel **5.7** dan Gambar **5.4** dapat dilihat bahwa 1-gram memiliki tingkat false positive paling rendah, dan 2-gram memiliki tingkat false positive yang sangat tinggi. 3-gram dan 4-gram memiliki akurasi yang tidak jauh berbeda dalam membedakan paket – paket bersih.

5.3.3 Skenario Pengujian True Positive

Skenario pengujian true positive aplikasi ini dapat dilihat pada Tabel **5.8**.

Tabel 5.8 Tabel Skenario Pengujian True Positive

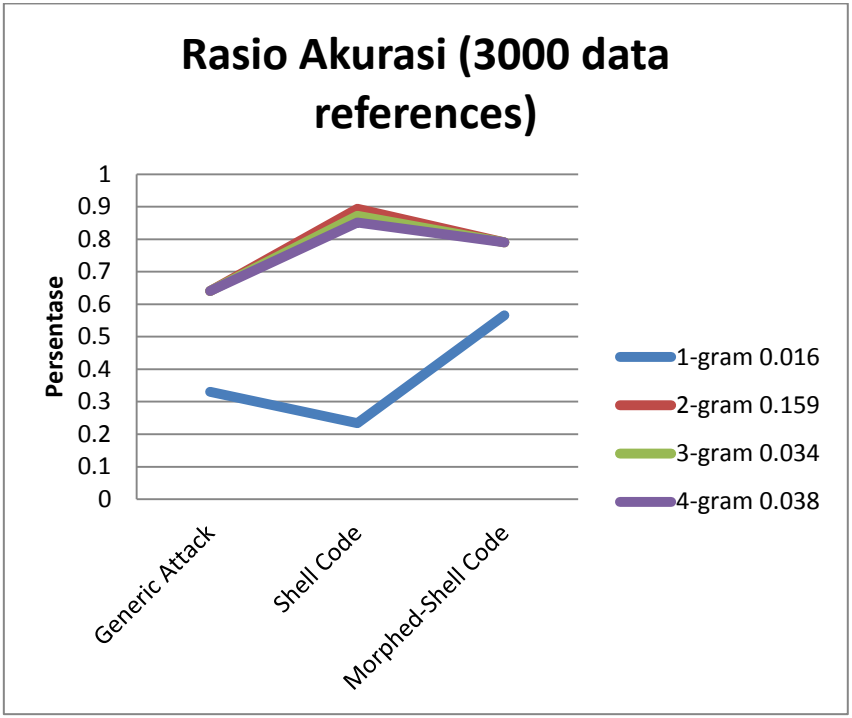
Deskripis	Bertujuan untuk mengetahui rasio true positive dari aplikasi ini
Langkah Pengujian	1. Pilih file yang berisi paket – paket serangan. 2. Jalankan program dan pilih n-gram yang dipakai untuk pengujian 3. Biarkan program berjalan hingga semua paket yang ada pada file sudah dianalisa 4. Catat rasio *true positive*-nya

5.3.4 Hasil Pengujian True Positive

Hasil pengujian ini adalah untuk mengetahui n-gram mana yang paling efektif dalam mendeteksi paket – paket serangan. Hasil pengujian true positive ini dapat dilihat pada Tabel **5.9** dan Gambar **5.5**.

Tabel 5.9 Tabel Hasil Pengujian True Positive (presisi 3 digit)

n-gram	True Positive		
	Generic Attack	Shell Code	Morphed-Shell Code
1-gram	0.330	0.234	0.565
2-gram	0.640	0.893	0.790
3-gram	0.640	0.872	0.790
4-gram	0.640	0.851	0.790



Gambar 5.5 Grafik Pengujian Akurasi

Pada Tabel **5.9** dan Gambar **5.5** dapat dilihat bahwa tingkat akurasi 1-gram sangatlah kecil dibandingkan dengan n-gram lainnya. Pada tingkat akurasi *true positive* ini, tingkat akurasi 2-gram selalu paling tinggi, terbalik dengan tingkat akurasi *false positive*-nya. Tingkat akurasi 3-gram dan 4-gram tidak memiliki tingkat perbedaan yang sangat signifikan.

5.4 Evaluasi

Dari uji coba yang telah dilakukan, diketahui bahwa aplikasi telah dapat bekerja dengan baik dan benar dalam menjalankan fungsionalitasnya. Evaluasi pengujian pada masing – masing pengujian dijelaskan sebagai berikut.

1. Pengujian run time untuk aplikasi sudah berjalan dengan cukup baik, namun dirasa masih kurang optimal. Untuk nilai n yang bernilai 1 dan 1000 data referensi ,aplikasi membutuhkan waktu 1000 detik untuk menganalisa 1000 paket. Namun pada saat aplikasi menggunakan n = 4, aplikasi membutuhkan 2200 detik. Angka ini lebih dari 2x lipat waktu 1-gram. Sedangkan untuk data referensi 5000, 1 gram membutuhkan waktu 5000 detik, sedangkan 4 – gram membutuhkan waktu 11000 detik. Dari sini dapat dilihat bahwa menggunakan 4-gram bukanlah cara yang optimal dalam penggunaan aplikasi ini.
2. Pengujian Penggunaan CPU dan penggunaan memori untuk aplikasi sudah berjalan dengan cukup baik. Dapat dilihat dari hasil pengujian bahwa jumlah n pada n-gram dan jumlah data referensi tidak berpengaruh banyak terhadap penggunaan CPU dan penggunaan memori
3. Pengujian False positive pada aplikasi berjalan dengan baik. Dari hasilnya dapat dilihat bahwa 1-gram memiliki tingkat rasio *false positive* terendah yang diikuti oleh 3-gram, 4-

gram, dan terakhir oleh 2-gram. Disini dapat dilihat bahwa 2-gram memiliki tingkat false positive yang sangat tinggi senilai 15,9% dari 1000 data yang dianalisa. 3-gram dan 4-gram juga tidak terlalu jauh berbeda dengan hasil dari 1-gram.

4. Pengujian *True Positive* pada aplikasi berjalan dengan baik. Disini, kebalikan dengan hasil uji *false positive*, 2-gram memiliki hasil terbaik untuk semua jenis serangan yang diuji. Di pengujian ini dapat dilihat bahwa 1-gram memiliki nilai *true positive* yang sangat rendah dibandingkan dengan n-gram lainnya, terutama pada pendeteksian shellcode attack yang hanya memiliki tingkat akurasi 23,4%. 3-gram dan 4-gram juga tidak beberda terlalu jauh dengan hasil dari 2-gram.

Secara keseluruhan , aplikasi ini dapat menjalankan fungsi – fungsinya dengan baik, yaitu mengklasifikasikan data bersih dan data kotor dari paket – paket yang dianalisa. Dapat dilihat bahwa untuk waktu run time, jumlah n-gram dan data referensi berpengaruh besar dalam waktu run time yang digunakan untuk menganalisa paket.

(Halaman ini sengaja dikosongkan)

BAB VI
PENUTUP

Dalam Bab terakhir ini dijelaskan kesimpulan yang didapat dari pengerjaan Tugas Akhir ini beserta saran – saran yang dapat dipertimbangkan untuk pengembangan atau penelitian lebih lanjut.

6.1 Kesimpulan

Berdasarakan hasil uji coba yang telah dilakukan , terdapat beberapa kesimpulan yang bisa diambil, yaitu:

1. Penggunaan metode *pearson chi-square test* dalam pendeteksian serangan memiliki tingkat akurasi rata – rata 0.6700289 untuk 3 jenis serangan yang diuji coba.
2. Penggunaan *library* WinPcap untuk mengambil n-gram dan menganalisis paket menghasil tingkat akurasi 0.6700289 untuk 3 jenis serangan yang diuji..
3. Metode *chi-square distance* diimplementasikan terhadap pola – pola n-gram yang muncul pada paket, menghasilkan tingkat akurasi rata – rata 0.6700289 untuk 3 jenis serangan dan tingkat akurasi false positive senilai 0.06175 untuk 4 jenis n-gram.
4. Metode 3 – gram adalah metode terbaik untuk digunakan karena tingkat *akurasi false positive* yang rendah senilai 0.034, dan tingkat rasio *true positive* yang tinggi senilai 0.767838 untuk 3 jenis serangan yang diuji , dan juga *running time* yang tidak terlalu tinggi sebanyak 4 detik tiap paket dengan menggunakan 3000 paket referensi.

6.2 Saran

Adapun saran yang dapat dipertimbangkan untuk pengembangan atau penelitian lebih lanjut adalah sebagai berikut :

1. Mencoba menggunakan perhitungan jarak yang lain seperti ad-hoc chi square distance atau metode *pattern counting.*
2. Menggunakan data referensi yang lebih baik lagi.
3. Menyimpan hasil perhitungan nilai normalisasi data referensi sebelumnya agar pada saat penentuan serangan tinggal dipanggil saja nilainya, sehingga mengurangi waktu *run time* dalam menganalisa paket.

Daftar Pustaka

[1] "Symantec Internet Security Threat Report, Vol. 17," [Online]. Available: http://www.symantec.com/threatreport/.

[2] "https://docs.microsoft.com/en-us/visualstudio/welcome-to-visual-studio".

[3] A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig, "Syntactic clustering of the web," vol. 29, no. 8-13, 1997.

[4] "http://www.ling.upenn.edu/~clight/chisquared.htm," [Online]. Available: http://www.ling.upenn.edu/~clight/chisquared.htm. [Accessed 3 July 2015].

[5] G. W. Snedecor and W. G. Cochran, Statistical Method, Eigth Edition, Iowa State University Press, 1989.

[6] "http://winpcap.software.informer.com/," [Online]. Available: http://winpcap.software.informer.com/. [Accessed 3 July 2015].

[7] B. Stroustrup, The C++ Programming Language (Third ed.), 1997.

[8] "DARPA '99 Dataset," [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1999data.html.

[9] R. Perdisci, "McPAD datasets".

[10] I. H. Ingham K., "Comparing anomaly detection technique for HTTP," in *Proceedings of the 10th international conference on recent advantages in intrusion detection*, 2007.

[11] T. R. G. G. Ariu D, "HMMLPayl : an intrusion detection system based on Hidden Markov models.," no. 221 - 41, 2011.

[12] "CLET polymorphic shellcode engine," [Online]. Available: http://www.phrack.org/issues.html?issue=61&id=9.

(Halaman ini sengaja dikosongkan)

Lampiran

```
#include "stdafx.h"
#include<iostream>
#include <stdio.h>
#include <cstdio>
#include <string>
#include <string.h>
#include <vector>
#include <fstream>
#include "pcap.h"
#include <iomanip>
#include <random>
#include <time.h>
#include <thread>
#include <future>
#include <Windows.h>
using namespace std;

void capture();// Fungsi untuk melakukan penangkapan paket – paket yang masuk.

void packet_handler(u_char *param, const struct pcap_pkthdr *header, const u_char *pkt_data);//fungsi untuk menyimpan paket – paket yang masuk ke dalam sebuah file.

int hextodec(string hexanumber);//fungsi untuk mengganti nilai hexadecimal menjadi nilai decimal.

double normalisasi(int frekuensi, int total);//Fungsi untuk menghitung nilai normalisasi dari pola

int detection(string attackfile,vector<double> normal,double chisqrval,int b, int c);//Fungsi untuk mendeteksi apakah paket yang
```

```
dianalisa berbahaya atau tidak

int glodet = 0;// Flag untuk menentukan apakah paket yang sedang dianalisa berbahaya atau tidak.

void analisis();//Fungsi untuk melakukan analisis paket berbahaya atau tidak.

int main()
{
	SetConsoleTitle(TEXT("N-Gram Analysis"));
	int i;
	cout<<"Pilih fitur yang akan digunakan : "<<endl<<"1. Menangkap file"<<endl<<"2. Menganalisa file"<<endl;
	cin >> i;
	if(i==1)
	{
		capture();
	}
	else if(i==2)
	{
		analisis();
	}
	else
	{
		exit(EXIT_FAILURE);
	}
	return 0;
}

void capture()
{
	string file;//variabel penyimpan lokasi dari file hasil capture aplikasi
```

```
	pcap_if_t *alldevs; //Container untuk semua nama dalam list interface
	pcap_if_t *d;
	pcap_t *adhandle; // Pendeskripsi untuk fungsi penangkapan paket
	pcap_dumper_t *dumpfile;//Pendeskripsi untuk lokasi file penyimpanan hasil penangkapan dari WinpCap

	int inum;
	int i=0;
	unsigned int netmask;

	char errbuf[PCAP_ERRBUF_SIZE];//Array char untuk menyimpan error

	fstream capturing;

	struct bpf_program fcode;

		if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1) // Melakukan pengecekan apakah ada perangkat yang dapat digunakan atau tidak
		{
			cout <<stderr<<"Error in pcap_findalldevs: "<<errbuf<<endl;
			exit(EXIT_FAILURE);
		}


	  for(d=alldevs; d; d=d->next) // Menunjukkan semua perangkat yang tersedia
		{
			cout<<++i<<". "<<d->name;
			if (d->description)
				cout<<" "<<d->description<<endl;
```

```
                else
                        cout<<"  tidak  ada
perangkat"<<endl;
        }

        if(i==0)//Jika tidak ada perangkat yang terdeteksi
akan menghentikan program
        {
                exit(EXIT_FAILURE);
        }

        cout<<"Pilih perangkat mana yang akan digunakan
(1 - "<< i << ") : ";
        cin >> inum;

        if(inum < 1 || inum > i)//Jika angka yang
dimasukkan tidak tersedia akan menghentikan program
        {
                printf("\nInterface number out of range.\n");
          pcap_freealldevs(alldevs);
                exit(EXIT_FAILURE);
        }

    for(d=alldevs, i=0; i< inum-1 ;d=d->next,
i++);//menginiliasasi perangkat yang akan digunakan sesuai dengan
apa yang kita pilih tadi.


        if ( (adhandle= pcap_open(d->name,  // Nama
perangkat
65536,  // Ukuran dari paketyang ditangkap
       // 65536 digunakan agar aplikasi dapat menangkap
seluruh paket pada link layers
PCAP_OPENFLAG_PROMISCUOUS,    // mode prosmicious
1000,  // read timeout
```

```
NULL,           // autentikasi untuk perangkat
errbuf          // buffer untuk error
                                                        ) )
== NULL)// Untuk menyiapkan perangkat yang dipilih agar siap untuk melakukan capture dan menghentikan program jika perangkat yang dipilih tidak bisa melakukan capture
                {
                        cout<<stderr <<endl<<"Perangkat tidak dapat dibuka "<<endl;
                   pcap_freealldevs(alldevs);
                        exit(EXIT_FAILURE);
                }


                cout <<"Masukkan Lokasi file hasil penangkapan : ";
                cin >> file;

                capturing.open(file, ifstream::out | ifstream::trunc);//Untuk membuat file jika file tersebut tidak ada sebelumnya, atau menghapus semua isi file jika file tersebut sudah ada
                capturing.close();

                dumpfile = pcap_dump_open(adhandle, file.c_str());//Menandakan file mana yang akan digunakan sebagai file hasil capture aplikasi

                if(dumpfile==NULL)
                {
                        fprintf(stderr,"\nFile penyimpanan tidak dapat dibuka.\n");
                        exit(EXIT_FAILURE);
                }
```

```
		printf("\nlistening on %s... Press Ctrl+C to stop...\n", d->description);


		pcap_freealldevs(alldevs);// Membebaskan semua perangkat lain yang tidak digunakan


		pcap_loop(adhandle, 0, packet_handler, (unsigned char *)dumpfile);//Fungsi untuk melakukan penangkapan secara berulang – ulang hingga program dihentikan
	}


void packet_handler(u_char *dumpfile, const struct pcap_pkthdr *header, const u_char *pkt_data)
{
	pcap_dump(dumpfile, header, pkt_data);//Untuk menyimpan paket – paket yang ditangkap ke dalam file.
}




void analisis()
{

	//for(int y=2;y<5;y++)
	//{

	//for(int z=2;z<5;z++)
	//{

	clock_t t1,t2;//Untuk container lama waktu program berjalan
```

```
        int b;//number of n in n-gram
        int c;//Panjang vector PatternSize
        int d;//variabel untuk perulangan
        int e;//nomor pola
        int f;//nomor flag
        int totfrek;//Jumlah frekuensi byte data yang dianalisa
        int totfrekA;//jumlah frekuensi byte data serangan
        int packetnum;//packet number
        int patternlength;//data input length
        int m;//flag to change file destination
        int fp=0;//jumlah serangan yang dideteksi
        //int y=1;
        //int n=1;
        //int j=1;
        //int tanda=1;
        int tes;//sebagai container data[i]
        int main;//container untuk hasil dari analisa attack1.txt
        int fut1;//untuk memanggil hasil dari async ret1
        int fut2;//untuk memanggil hasil dari async ret2
        int fut3;//untuk memanggil hasil dari async ret3
        int fut4;

      double falsepos;//menunjukkan persentase serangan yang
dideteksi dari paket - paket yang dianalisa
        double seconds;//second container
        double chisqrval;//value of chisqr
        double value;//calculated chisqr between analyzed data and
dataset


        //string tesfile ;
        //string tesfile1;
        //string tesfile2;
```

```
        string file;//container untuk file yang ingin dianalisa
        string line;//For Getting line from text file
        string cek;//For Byte frequency checks
        string output, input;//output dari file stream
        //string unigram;
        //string bigram;
        //string trigram;
        //string quadgram;
        //string pentagram;
        //string source;
        //string prime;
        //string secondary;


        fstream myfile;
        fstream fileku;
        fstream temporary;//Stream Analisis.txt
        fstream chisqr;//stream chisqr.txt


        future<int> ret1;//container async attack2.txt
        future<int> ret2;//container async attack3.txt
        future<int> ret3;//container async attack4.txt


        pcap_t * pcap;
        char errbuff[PCAP_ERRBUF_SIZE];
        struct pcap_pkthdr *header;
        const unsigned char *data;
        unsigned int packetCount = 0;

        cout.precision(3); // Untuk menset presisis dari cout menjadi
3 angka dibelakang koma

        cout<<"Pilih n dari n-gram yang ingin digunakan(1,2,3,4) :
```

```
“<<;
        cin>>b;

        c=pow(16,b);//jumlah pola yang mungkin muncul dari n-gram

        if(0>b>5)
        {
                printf("Not supported.");
                exit(EXIT_FAILURE);
        }

        cout << " Masukkan location lengkap dari file yang ingin di analisa : " ;
        cin >> file;

        if(pcap_open_offline(file.c_str(), errbuff))//untuk mencek apakah ada file hasil capture dengan nama file
        {
                pcap = pcap_open_offline(file.c_str(), errbuff);
        }
        else
        {
                cout << "file not found.";
                exit(EXIT_FAILURE);
        }

        t2=clock();
         fileku <<endl<<"Source file : " << source<<endl<<endl;

while (int returnValue = pcap_next_ex(pcap, &header, &data) >= 0)//melakukan perulangan selama masih ada data pada file yang kita sediakan
        {
                vector<int> PatternSize(c);//Frekuensi dari Pola c
```

```
		vector<int> flag(c);//flag pola mana saja yang muncul untuk data yang dianalisa
		vector<double> normal(c);//normalisasi dari data yang dianalisa



		myfile.open(“Serangan Terdeteksi.txt” , ios_base::app);
		chisqr.open("chisqr.txt");
		packetCount++;
		cout<<"Packet # " << packetCount <<endl;//Menunjukkan paket nomor berapa yang sedang dianalisa

		temporary.open ("Analisis.txt", ifstream::out | ifstream::trunc);//untuk menclear file analisis

		temporary.close();

		temporary.open("Analisis.txt", ios_base::app);//mempersiapkan file untuk ditulis

	for (unsigned int i=0; (i < header->caplen ) ; i++)//memasukkan nilai hex dari paket yang ingin dianalisa dari file pcap
		{
			tes = data[i];

			temporary<< hex <<setfill('0') << setw(2)<<tes;		// menuliskan tes sebagai hex 2 digit melalui stream temporary
	}
		temporary.close();
		temporary.open("Analisis.txt");
		temporary >> output;
		patternlength = output.length();
		temporary.close();
```

```
totfrek = 0;
f=0;

for(d=0;d<=patternlength-b; d++)//Mencari frekuensi pola dan pola apa saja yang ada pada paket
{
    e = hextodec(output.substr(d,b));

    if(!PatternSize[e])//Memberi flag pada pola yang ada pada paket
    {
        flag[f] = e;
        f++;
    }

    PatternSize[e]++;
    totfrek++;
}

f--;

for(d=0;d<f-1; d++)//mencari nilai chisqr dari degree of freedom(f)
{
    chisqr >> input;
    chisqr >> chisqrval;
}

chisqr.close();

for(d=0; d<=f; d++)//normalisasi dari paket yang ingin dianalisa
{
    normal[flag[d]] = normalisasi(PatternSize[flag[d]], totfrek);
```

```
                }

                ret1 =
async(detection,"Benign30002.txt",normal,chisqrval,b,c);//thread
untuk membandingkan paket masuk dan paket dataset pada
attack2.txt . Akan mengembalikan nilai 1 apabila terdeteksi
serangan.

                ret2 =
async(detection,"Benign30003.txt",normal,chisqrval,b,c);//thread
untuk membandingkan paket masuk dan paket dataset pada
attack3.txt . Akan mengembalikan nilai 1 apabila terdeteksi
serangan.

                ret3 =
async(detection,"Benign30004.txt",normal,chisqrval,b,c);  //thread
untuk membandingkan paket masuk dan paket dataset pada
attack4.txt . Akan mengembalikan nilai 1 apabila terdeteksi
serangan.

                main =
detection("Benign30001.txt",normal,chisqrval,b,c); //fungsi untuk
membandingkan paket masuk dan paket dataset pada attack1.txt .
Akan mengembalikan nilai 1 apabila terdeteksi serangan.

                fut1 = ret1.get();// Meangambil nilai yang dihasilkan
oleh tread ret1.
                fut2 = ret2.get();// Meangambil nilai yang dihasilkan
oleh tread ret1.
                fut3 = ret3.get();// Meangambil nilai yang dihasilkan
oleh tread ret1.
                //fut4 = ret4.get();

                if(fut1+fut2+main+fut3)//apabila salah satu dari
thread mengembalikan 1, berarti paket yang dianalisa termasuk
```

```
serangan
                {
                        myfile<<" Packet # "<<packetCount<<" is
an attack"<<endl;
                        fp++;
                }

                t1=clock();
                t1=t1-t2;
                seconds = t1/CLOCKS_PER_SEC;

                //cout<<endl<<"time : " << seconds<<" false
positive = " << (double)fp/packetCount<<" ";//untuk mengetahui
berapa lama program berjalan dan nilai false positivenya.

                glodet=0;

                myfile.close();
        }
}

//Fungsi untuk mencari kesamaan dari paket yang dianalisa dengan
paket dataset yang sudah tersedia
int detection(string attackfile,vector<double> normal,double
chisqrval,int b, int c)
{
                fstream fileku;
                fileku.open(attackfile);

                string input;
                string tes;

                int detect;//penanda apakah paket adalah serangan
atau bukan
                int patternlength;//panjang dari string input
```

```
int totfrekA;//total frekuensi paket
int d;//variabel untuk perulangan
int e;//container untuk bentuk pola
int p;//jumlah pola unik yang muncul
double value;//hasil dari euclidean distance
int loopflag=1;//flag untuk menghentikan perulangan jika kondisi sudah terpenuhi
int z;

if(fileku.is_open())
{

	z=0;

	while(loopflag)//loop untuk membandingkan semua paket yang ada di dataset dan paket yang dianalisa
	{

		if(glodet)
		{
			return glodet;
		}

		if(!getline(fileku,input))//untuk menghentikan while loop apabila menemui end of file
		{
			loopflag =0;
			fileku.close();
			//detect=0;
			z=0;
		}

		vector<int> PatternSizeA(c);//frekuensi dari pola
```

```
                    vector<int>          flagA(c);//flag
untukpola yang muncul
                    vector<double>  normalA(c);//hasil
normalisasi dari pola
                    value = 0;
                    totfrekA = 0;
                    p=0;
                    fileku>>input;
                    patternlength = input.length();

                    for(d=0;d<=patternlength-b;
d++)//mencari frekuensi pola dari paket dan memberi flag pada pola
yang ada
                    {
                        e                    =
hextodec(input.substr(d,b));

                        if(!PatternSizeA[e])// untuk
memberi flag pada pola yang ada di paket
                        {
                            flagA[p] = e;
                            p++;
                        }

                        PatternSizeA[e]++;
                        totfrekA++;
                    }

                    p--;
                    double calculation;

                    for(d=0;  d<=p;  d++)//menghitung
nilai normalisasi paket dari dataset dan nilai kemiripan paket yang
dianalisa dan paket dataset
                    {
```

```
                                        normalA[flagA[d]]       =
normalisasi(PatternSizeA[flagA[d]],totfrekA);
                                        calculation             =
pow((normal[flagA[d]] - normalA[flagA[d]]),2)/normalA[flagA[d]];
                                        value+=calculation;
                                }

                                //cout<<"value : " <<value;

                                if(value>chisqrval)//kemiripan
antara paket yang dianalisa dan paket dataset
                                {
                                        //detect=1;
                                        fileku.close();
                                        loopflag=0;
                                        glodet=1;
                                }

                        }

                }

                return z;
}



//fungsi untuk mencari nilai normalisasi dari data
double normalisasi(int frekuensi, int total)
{
        double normal;
        normal = (double)frekuensi/total;
        return normal;
}
```

```
//fungsi untuk mendapatkan nilai decimal dari hex
int hextodec(string hexanumber)
{
	int temp=0,a=0;
	int length = hexanumber.length();

	for(int e=0; e<length;)
	{

		if (hexanumber[e] >= 48 && hexanumber[e] <= 57)
// adalah *hexstr antara 0-9
		{
			temp += (((int)(hexanumber[e])) - 48) *
pow(16, length - e - 1);
		}

		else if ((hexanumber[e] >= 65 && hexanumber[e]
<= 70))   // adalah *hexstr antara A-F
		{
			temp += (((int)(hexanumber[e])) - 55) *
pow(16, length - e - 1);
		}

		else if (hexanumber[e] >= 97 && hexanumber[e]
<= 102)   // adalah *hexstr antara a-f
		{
			temp += (((int)(hexanumber[e])) - 87) *
pow(16, length - e - 1);
		}

		e++;
```

```
        }

        return temp;
}
```

Biodata Penulis

Reinhard Ruben Rumare, lahir di kota Bandung, pada tanggal 14 Oktober 1993. Penulis adalah anak kedua dari tiga bersaudara. Pendidikan formal yang telah ditempuh penulis adalah TK Santa Maria(1998 – 1999), SD Methodist 1(1999 - 2005), SMP St. Thomas 1(2005-2008), SMA St. Thomas 1(2008-2011). Setelah lulus dari SMA St. Thomas 1, penulis diterima di jurusan Teknik Informatika ITS angkatan 2011 dengan NRP 5111100110. Penulis dapat dihubungi melalui alamat e-mail reinhardruben@gmail.com.