DETEKSI PROBING OS FINGERPRINTING MENGGUNAKAN ALGORITMA NAIVE BAYES

SKRIPSI

Diajukan untuk memenuhi persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:

AMRI WAHYU JADHI PRATAMA

NIM. 0810680020

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

LEMBAR PERSETUJUAN

DETEKSI PROBING OS FINGERPRINTING MENGGUNAKAN ALGORITMA NAIVE BAYES

SKRIPSI

Diajukan untuk memenuhi persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

AMRI WAHYU JADHI PRATAMA

NIM. 0810680020

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eko Sakti P., S.Kom., M.Kom.	**Novanto Yudistira, S.Kom., M.Sc.**
NIK. 860805 06 1 1 0252	**NIK. 831110 16 1 1 0425**

LEMBAR PENGESAHAN

DETEKSI PROBING OS FINGERPRINTING
MENGGUNAKAN ALGORITMA NAIVE BAYES

SKRIPSI

Diajukan untuk memenuhi persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

Amri Wahyu Jadhi Pratama
NIM. 0810680020

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 18 Juli 2013

Penguji I	Penguji II
Kasyful Amron, ST., M.Sc	**Aswin Suharsono, ST., MT**
NIP. 19750803 200312 1 003	**NIP. 840919 06 11 0251**

Penguji III

Gembong Edhi S., ST., MT
NIK. 850920 16 1 1 0373

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, MT
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 Juli 2013

Mahasiswa,

Amri Wahyu Jadhi Pratama

NIM.0810680020

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul "Deteksi Probing OS *Fingerprinting* Menggunakan Algoritma Naive Bayes" dengan baik.

Penulisan laporan skripsi ini diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer dan merupakan salah satu syarat yang harus ditempuh oleh setiap mahasiswa Teknik Informatika Universitas Brawijaya Malang.

Penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu, diantaranya :

1. Bapak Eko Sakti Pramukantoro., S.Kom., M.Kom. dan Bapak Novanto Yudistira, S.Kom., M.Sc. selaku pembimbing dalam menyelesaikan laporan skripsi ini.
2. Orang tua yang telah memberikan dukungan moral dan material.
3. Teman teman yang telah membantu memberi saran dan kritik atas laporan ini.

Penulis menyadari bahwa laporan skripsi ini masih banyak kekurangan dan jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga laporan skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa Teknik Informatika Universitas Brawijaya.

Malang, 24 Juli 2013

Penulis

ABSTRAK

Amri Wahyu Jadhi Pratama. 2013. Deteksi Probing OS *Fingerprinting* Menggunakan Algoritma Naïve Bayes. Skripsi Program Studi Teknik Informatika, Fakultas Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Pembimbing: Eko Sakti Pramukantoro., S.Kom., M.Kom. dan Novanto Yudistira, S.Kom., M.Sc.

Probing OS *Fingerprinting* merupakan langkah awal dari sebuah intrusi yang bertujuan untuk mendapatkan informasi sistem operasi yang digunakan oleh target. Informasi tersebut digunakan penyerang untuk memilih *exploit* yang tepat. Metode pencegahan yang sering diterapkan saat ini adalah *firewall* dan *rule based* IDS. Namun *firewall* hanya dapat melakukan langkah pertahanan pasif sedangkan *rule based* IDS bergantung pada aturan yang harus selalu diperbarui. Untuk itu diperlukan sistem yang dapat melakukan langkah pertahanan aktif serta tidak bergantung pada rules yaitu *anomaly based intrusion detection system*. Penulis mengimplementasikan IDS menggunakan algoritma mesin pembelajaran Naïve Bayes dan menguji pada jaringan lokal. Hasil pengujian menunjukkan anomaly based IDS dapat mendeteksi probing OS *fingerprinting* dengan tingkat akurasi mencapai 96% dan mencegah akses lebih lanjut dari intruder.

Kata Kunci: IDS, Data Mining, Probing

ABSTRACT

Amri Wahyu Jadhi Pratama. 2013. Deteksi Probing OS *Fingerprinting* Menggunakan Algoritma Naïve Bayes. Skripsi Program Studi Teknik Informatika, Fakultas Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Pembimbing: Eko Sakti Pramukantoro., S.Kom., M.Kom. dan Novanto Yudistira, S.Kom., M.Sc.

Probing OS Fingerprinting is common initial step from intrusion that has purpose to get information about target operating system. Intruder uses information to choose right exploit. Prevention methods commonly used are firewall and rule based IDS. However firewall only can do passive defense meanwhile rule based IDS depends on rules which always updated. So we need system can do active defense and doesn't depend on rules which is anomaly based intrusion detection system. We apply IDS that using Naïve Bayes machine learning algorithm and test it on Local Area Network (LAN). Result of the test shows anomaly based IDS can detect OS fingerprinting probing with high accuracy 96% and prevent intruder to access further.

Keyword : IDS, Data Mining, Probing

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dengan perkembangan teknologi informasi dan telekomunikasi saat ini, pengguna mulai sadar akan resiko keamanan komputer. Menurut Garfinkel dan Spafford, komputer dikatakan aman jika bisa diandalkan dan perangkat lunaknya bekerja sesuai dengan yang diharapkan. Keamanan komputer memiliki 5 aspek antara lain *Availability* (kemampuan untuk mendapatkan informasi), *Confidentiality* (kerahasiaan informasi), *Data Integrity* (keabsahan data), *Control*, dan *Audit* [SIM-05]. Untuk mendukung keamanan tersebut dibutuhkan mekanisme pertahanan. Mekanisme yang diterapkan untuk membuat jaringan komputer lebih aman antara lain *firewall* dan *intrusion detection system* (IDS).

Namun *firewall* yang dianggap metode untuk melindungi keamanan informasi hanya dapat melakukan langkah pertahanan pasif. Meskipun *firewal* dapat melindungi permintaan akses yang tak terduga dari luar, tapi tidak dapat memeriksa apakah aliran data yang lewat mempunyai kode berbahaya atau tidak. Dengan beragam metode dan alat serangan, *firewall* sendiri tidak dapat memenuhi tuntutan keamanan. Untuk itulah dikembangkan *intrusion detection system.*

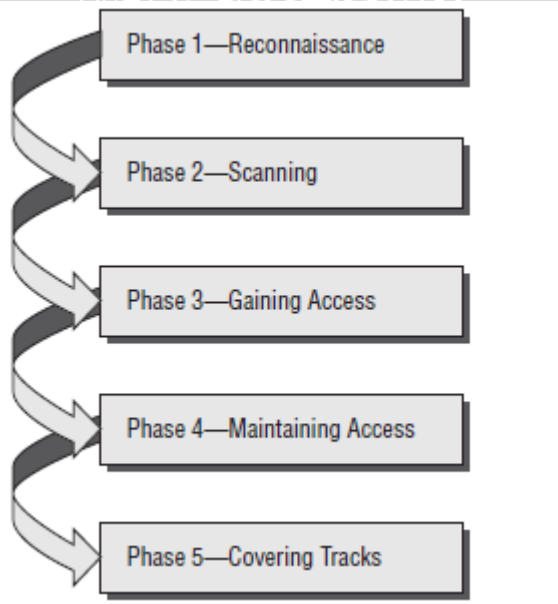
Menurut Song Naiping dan Zhou Genyuan, *Intrusion Detection System* (IDS) merupakan sistem yang dapat mendeteksi dan mengidentifikasi serangan berbahaya pada sistem komputer dan sistem jaringan dengan mengumpulkan dan menganalisa data, menemukan perilaku penyerangan yang tidak biasa atau peristiwa yang tidak biasa dan mengambil langkah-langkah tertentu untuk mencegah serangan sehingga mengurangi jumlah kerugian [NAI-10].

Berdasarkan teknologi yang digunakan, *intrusion detection* dibagi menjadi dua tipe, *misuse detection* dan *anomaly detection* [BOL-09]. *Misuse detection* mengumpulkan informasi, kemudian membandingkan dengan pola serangan yang telah diketahui. Keuntungan dari *misuse detection* adalah akurasi yang tinggi. Sedangkan *anomaly detection* menggunakan pendekatan bahwa serangan merupakan aktifitas yang berbeda dari aktifitas normal. Sistem melakukan pembelajaran ketika sistem berjalan pada aktifitas normal dan menggunakannya

sebagai dasar untuk menemukan perilaku yang berbeda dari pola aktifitas normal. Berdasarkan lingkungan jaringannya, *intrusion detection system* dibagi menjadi dua tipe, *network IDS* dan *host IDS*. *Network IDS* ditempatkan pada titik tertentu di jaringan untuk mengawasi lalu lintas data yang menuju ataupun berasal dari semua *device*. Sedangkan *host IDS* merupakan aplikasi IDS yang terpasang langsung ke sebuah host di jaringan.

Sebagian besar *intrusion detection system* saat ini bergantung pada *signature* yang dibuat seperti halnya *anti virus* yang harus diperbarui terus menerus untuk dapat menangkal serangan baru secara efektif. Terdapat kebutuhan untuk fokus pada pendeteksian serangan yang tidak mengandalkan pendekatan berbasis *signature*. Hal ini merujuk pada pendekatan lain dari *intrusion detection* yang mendeteksi kelainan pada jaringan.

Kimberly Graves dalam *Certified Ethical Hacker Study Guide v6* berpendapat bahwa terdapat 5 tahapan dalam intrusi dan tahapan awal adalah mengumpulkan informasi dari target [GRA-10]. Fase tersebut dapat dilihat pada gambar 1.1. Informasi seperti alamat IP, sistem operasi, service yang berjalan dan port yang terbuka dapat membantu penyusup menentukan *exploit* yang digunakan untuk menyerang sistem. Pengumpulan informasi ini menggunakan probing untuk mengetahui informasi tersebut.



Gambar 1.1 Fase *Hacking*

Probe atau yang biasa disebut probing adalah suatu usaha untuk mendapatkan informasi tentang sistem. Probing tidak begitu berbahaya namun biasanya akan diikuti oleh tindakan lain yang membahayakan keamanan sistem. Metode ini merupakan langkah awal yang digunakan untuk melakukan intrusi. Salah satu tipe probing adalah OS *fingerprinting* yang berguna untuk mengetahui sistem operasi yang digunakan oleh target.

Ketika sistem operasi sudah diketahui, penyerang dapat menentukan *exploit* yang harus digunakan. Selain itu aplikasi atau *service* yang berjalan juga bisa ditebak. Dengan mendeteksi probing OS *fingerprinting* dan memblokir serangan dari awal, diharapkan tidak banyak informasi sistem yang diketahui dan mencegah penyerang menggunakan metode lain. Menurut Gordon Lyon, beberapa tipe serangan yang dapat digunakan ketika operating sistem telah diketahui adalah *buffer overflow* dan *format-string exploit* [LYO-07].

Pendeteksian probing saat ini dapat menggunakan aplikasi *intrusion detection system* yang dapat diunduh bebas di internet seperti Snort dan Morph [GRE-06]. Tetapi dalam pendeteksian serang masih bergantung pada *database signature*. Dengan menggunakan algoritma naïve bayes yang memiliki kecepatan pengolahan data dan tingkat deteksi yang tinggi, penulis membuat sebuah sistem pendeteksian serangan yang ditanam di setiap host jaringan sehingga pendeteksian probing tidak lagi bergantung pada *signature* dan dapat mendeteksi serangan probing yang belum diketahui.

1.2 Rumusan Masalah

Berdasarkan permasalahan yang telah dijelaskan pada bagian latar belakang, penulis merumuskan masalah sebagai berikut:

- Bagaimana mendapatkan data lalu lintas paket normal dan serangan probing OS *fingerprinting*?
- Bagaimana membuat pola serangan probing OS *fingerprinting*?
- Bagaimana melakukan klasifikasi serangan menggunakan algoritma naïve bayes ?
- Bagaimana melakukan training pada *anomaly-based intrusion detection system*?

- Bagaimana mengevaluasi *anomaly-based intrusion detection system* yang ditanam di target *host* ?

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus dan tidak semakin berkembang, maka penelitian menitikberatkan pada beberapa hal sebagai berikut:

- Penelitian ini mengimplementasikan *anomaly-based intrusion detection system* yang dipasang di host jaringan.
- Algoritma yang digunakan untuk klasifikasi serangan adalah Naïve Bayes.
- Serangan yang diteliti adalah probing OS *fingerprinting*.
- Platform yang digunakan untuk target adalah Linux Ubuntu 13.04 dan *host* adalah Backtrack 5 R2 dan Microsoft Windows.

1.4 Tujuan

Tujuan dari penelitian ini antara lain :

- Mampu membuat *anomaly-based intrusion detection system* untuk mendeteksi probing OS *fingerprinting* secara akurat.
- Mampu menutup koneksi dari pengguna lain yang dianggap mengirimkan probing OS *fingerprinting*.

1.5 Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini adalah sebagai berikut:

- Bagi Penulis
  - Mengaplikasikan ilmu yang diperoleh selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya.
  - Menambah pengetahuan tentang pembuatan dan implementasi *intrusion detection system* berdasarkan klasifikasi serangan.

- Bagi Mahasiswa
  - Menambah pengetahuan tentang penanggulangan serangan pada jaringan komputer.
  - Menjadi bahan referensi untuk meneliti serangan lain dengan menggunakan metode yang sama.

1.6 Sistematika Penulisan

Sistematika isi dan penulisan dalam skripsi ini antara lain :

Bab I Pendahuluan

Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat dari penelitian serta sistematika pembahasan.

Bab II Kajian Pustaka dan Dasar Teori

Berisi tentang teori dasar *intrusion detection system*, probing, tcpdump, dan klasifikasi.

Bab III Metodologi Penelitian dan Perancangan

Bab ini berisi tentang gambaran umum perancangan dan implementasi sistem, meliputi: studi literatur, analisis kebutuhan, perancangan sistem, implementasi perangkat sistem, pengujian sistem dan pengambilan kesimpulan dan saran.

Bab IV Implementasi

Membahas tentang implementasi dari sistem *anomaly-based intrusion detection system*.

Bab V Pengujian dan Analisis

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

Bab VII Penutup

Memuat kesimpulan yang diperoleh dari implementasi dan pengujian sistem serta saran – saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Penelitian Terkait

Salah satu penelitian tentang *intrusion detection system* dilakukan oleh Song Naiping dan Zhou Genyuan yang berjudul "A study on Intrusion Detection Based on Data Mining" pada tahun 2010. Penelitian tersebut menjelaskan adanya kelemahan pada sistem pendeteksi serangan yang banyak digunakan saat ini dan mengusulkan penggunaan data mining agar IDS dapat menghasilkan pendeteksian yang akurat pada data dengan jumlah yang besar. [NAI-10]

Penelitian lain yang membahas *intrusion detection system* berjudul "A Comparative Study of Data Mining Algorithms For Network Intrusion Detection" oleh Mrutyunjaya Panda dan Manas Ranja Patra pada tahun 2008. Fokus penelitian adalah membandingkan algoritma J48, ID3 dan Naïve Bayes. Hasil dari penelitian tersebut memperlihatkan bahwa algoritma Naïve Bayes memiliki tingkat kecepatan pengolahan data dan tingkat akurasi yang lebih baik jika dibandingkan dengan algoritma J48 dan ID3. [PAN-08]

2.2 Intrusion Detection System

Intrusion Detection System (IDS) merupakan sistem yang dapat mendeteksi dan mengidentifikasi serangan berbahaya pada sistem komputer dan sistem jaringan dengan mengumpulkan dan menganalisa data, menemukan perilaku penyerangan yang tidak biasa atau peristiwa yang tidak biasa dan mengambil langkah-langkah tertentu untuk mencegah serangan sehingga mengurangi jumlah kerugian[NAI-10].

2.2.1. Klasifikasi *Intrusion Detection System*

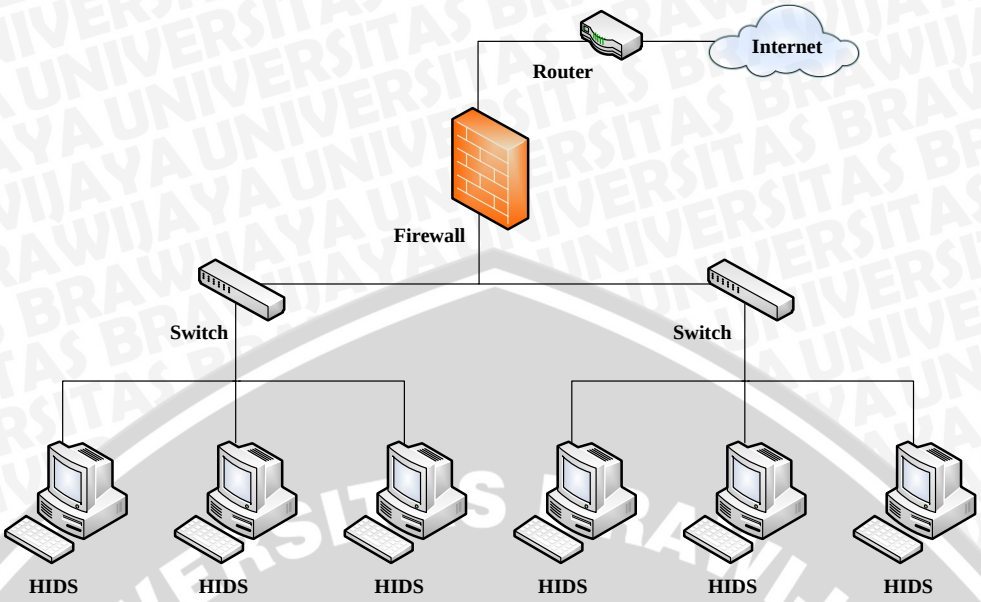
Berdasarkan lingkungan jaringan yang digunakan, terdapat dua tipe *intrusion detection system* diantaranya *host-based intrusion detection system* dan *network-based intrusion detection system*.

1. *Host-basedintrusion detection system*

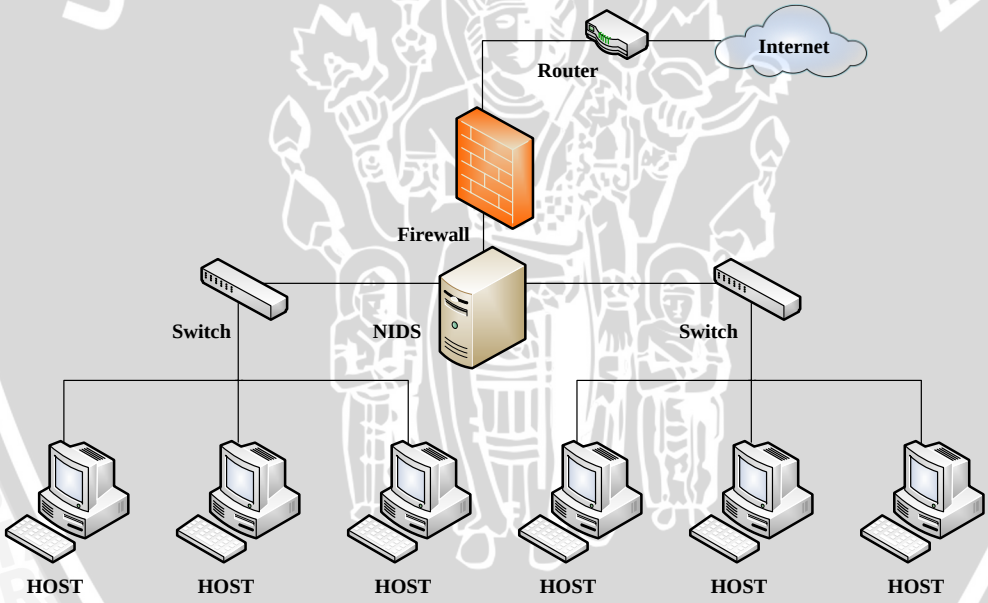
Host-based intrusion detection system atau yang biasa disebut HIDS terpasang di dalam host. IDS tipe ini dapat melihat ke dalam sistem dan file log aplikasi untuk mendeteksi aktifitas intrusi. Sistem ini bersifat reaktif, dalam artian IDS akan memberitahukan ketika sesuatu telah terjadi. Beberapa HIDS bersifat proaktif, dapat mengidentifikasi lalu lintas data di jaringan yang masuk ke dalam beberapa host dan mengirimkan *alert* setiap saat kepada pengguna. Penempatan HIDS pada topologi jaringan dapat dilihat pada gambar 2.1.

2. *Network-basedintrusion detection system*

NIDS merupakan *intrusion detection system* yang menangkap seluruh paket data yang melewati jaringan dan mencocokan dengan database *rule*. Semua data akan diperiksa oleh NIDS, namun cara ini dapat mengganggu kecepatan akses ke seluruh *host* di dalam jaringan. Penempatan NIDS pada jaringan dapat dilihat pada gambar 2.2.



Gambar 2.1 *Host* IDS



Gambar 2.2 *Network* IDS

Berdasarkan teknologi yang digunakan, *intrusion detection system* dapat dibagi menjadi dua kategori : *anomaly-based detection* dan *signature –based detection*.

1. *Anomaly-based detection*

Anomaly-based detection, juga disebut *activity-based detection*, mengacu pada pemantauan intrusi berdasarkan penyimpangan aktifitas pengguna atau penggunakan sumber daya dari kondisi normal. Ketika sistem bekerja, aplikasi *anomaly-based detection* menghasilkan kontur yang sedang aktif, membandingkan dengan kontur original dan memperbaharui kontur original pada saat yang sama. Ketika perilaku mencurigakan terjadi maka dianggap sebuah intrusi. *Behaviour-based detection* berbasis sistem independen sehingga dapat digunakan secara umum. IDS tipe ini mampu mendeteksi metode serangan yang belum pernah ada sebelumnya. Namun tidak dapat menggambarkan secara komprehensif perilaku setiap pengguna dalam sistem sehingga kelemahan utamanya adalah tingginya alaram *false negative*. Di sisi lain, rule perlu diperbaharui terus-menerus. Jika penyerang mengetahui sistem dalam pantauan *intrusion detection system*, mereka sengaja melatih *detection system* untuk beberapa waktu sehingga yang awalnya dianggap perilaku yang menyimpang dapat dianggap normal.

2. *Signature –based detection*

Dalam deteksi, *intrusion* menghasilkan model dan jejak yang tertinggal di sistem yang diamati merupakan dasar dari pengambilan keputusan. Oleh karena itu, *signature* dari aktifitas berbahaya dapat didefinisikan berdasarkan *rule* atau *expert knowledge*, dan membandingkan dengan peristiwa yang diamati sehingga dapat menilai apalah sistem telah diserang atau tidak. *Signature-based detection* dapat secara akurat mendeteksi beberapa serangan tetapi sistem tidak dapat mendeteksi serangan yang belum diketahui.

2.2.2. Komponen *Intrusion Detection System*

Intrusion detection system secara logikaldibagi menjadi beberapa komponen. Komponen tersebut bekerjsa sama untuk mendeteksi serangan tertentu dan menghasilkan peringatan kepada pengguna.

1. *Packet Decoder*

   *Packet Decoder*menangkap paket dari antarmuka jaringan dan mempersiapkan paket untuk *preprocessor* atau dikirim ke *detection engine.* Antarmuka jaringan yang digunakan biasanya ethernet, SLIP, PPP dan lain sebagainya.

2. *Preprocessor*

   Preprocessor merupakan bagian penting bagi IDS yang berfungsi untuk menyiapkan paket data untuk dianalisa oleh *detection engine*. *Preprocessor* melakukan konversi dan filterisasi paket yang telah ditangkap sehingga dapat dibaca oleh *detection engine.*

3. *Detection Engine*

   Komponen ini merupakan komponen terpenting dari IDS. *Detection engine* bertanggung jawab mendeteksi aktifitas intrusi yang terdapat pada paket data yang telah ditangkap.

4. *Logging and Alerting System*

   Komponen ini bertugas untuk menyimpan daftar serangan yang teridentifikasi dan memberi peringatan kepada pengguna.

5. *Output Modules*

   Fungsi dari modul ini dapat berbeda tergantung bagaimana pengguna ingin menyimpan keluaran yang dihasilkan oleh *logging andalerting system.* Pada dasarnya modul ini mengatur tipe keluaran yang dihasilkan oleh *logging andalerting system*

2.2.3. Alarm *Intrusion Detection System*

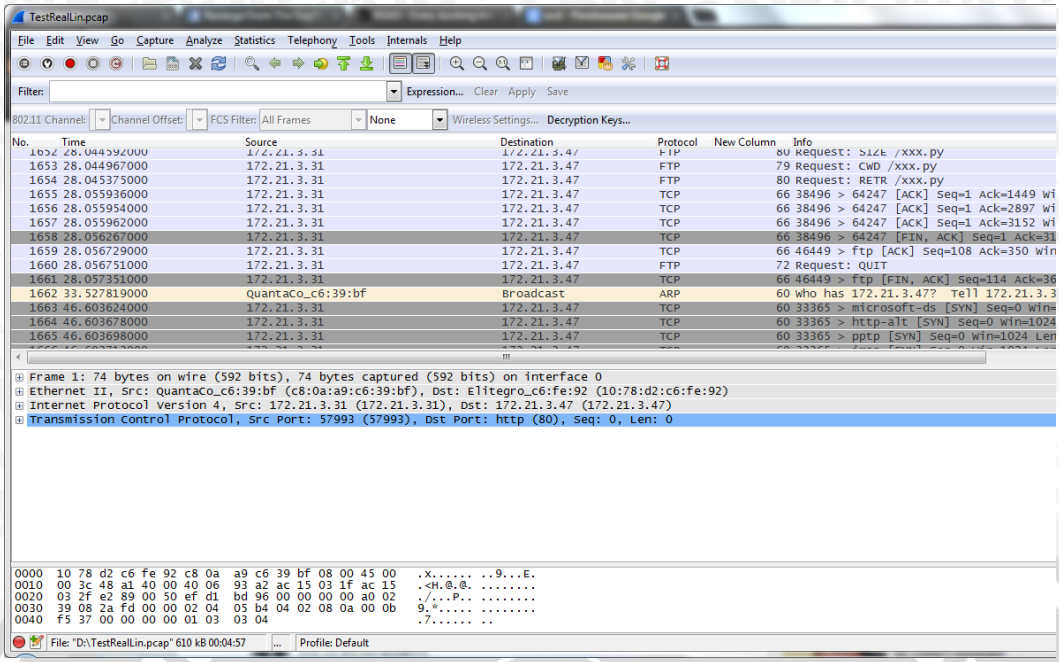
Akurasi dari sensor *intrusion detection system* akan mentrigger alarm sehingga dapat diketahui apakah terdapat intrusi atau tidak. Terdapat empat jenis alarm yang akan dihasilkan oleh *intrusion detction system* :

- *True Negative* (TN) : kondisi dimana lalu lintas data berjalan normal dan tidak ada alarm yang dibangkitkan.
- *True Positive* (TP) : kondisi dimana alarm akan ter-*trigger* jika ditemukan kecocokan yang diidentifikasi sebagai serangan.
- *False Negative* (FN) : kondisi dimana tidak ada alarm meski serangan telah masuk ke dalam sistem komputer.
- *False Positive* (FP) : kondisi dimana lalu lintas data berjalan normal namun ada alarm yang dibangkitkan. Saat ini banyak penelitian yang memfokuskan untuk mengurangi FP.

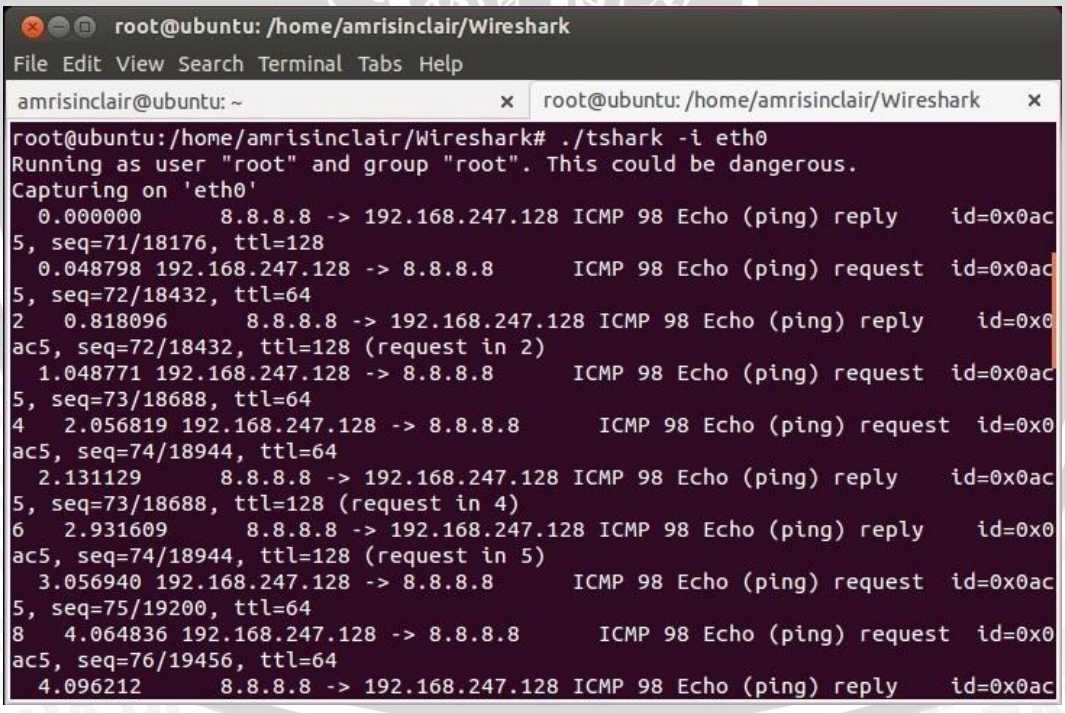
2.3 Wireshark/Tshark

Tshark merupakan aplikasi penangkap paket data yang umum digunakan yang berjalan pada terminal linux sedangkan versi GUI disebut dengan Wireshark. Tshark menampilkan paket TCP/IP dan paket lainnya yang dikirim atau diterima melalui jaringan komputer. Kedua aplikasi tersebut merupakan perangkat lunak bebas yang didistribusikan menggunakan lisensi GNU *General Public License*.

Tshark bekerja pada sistem operasi Unix seprti Linux, Solaris, BSD, Mac OS X, dan Windows. Pada sistem tersebut, Tshark menggunakan *library* libpcap untuk menangkap paket. Sedangkan pada sistem operasi Windows lebih dikenal dengan sebutan WinCap. Antarmuka aplikasi Wireshark dapat dilihat pada gambar 2.3 sedangkan tampilan antarmuka Tshark yang sedang berjalan dapat dilihat pada gambar 2.4.



Gambar 2.3 Aplikasi Wireshark



Gambar 2.4 Aplikasi Tshark

Penggunaan Tshark dalam penelitian ini dikarenakan kemudahan dan automatisasi sistem yang dibuat. Perintah untuk menjalankan Tshark ditempatkan pada sebuah bash file sehingga mempermudah pengguna ketika menangkap paket dan melakukan konversi file. Paket yang ditangkap disimpan dalam sebuah file

pcap dan selanjutnya dikonversi menjadi file csv sehingga dapat dibaca oleh aplikasi ataupun pengguna itu sendiri.

2.4 Probing

Probe atau yang biasa disebut probing adalah suatu usaha untuk mendapatkan informasi tentang sistem. Probing tidak begitu berbahaya namun biasanya akan diikuti oleh tindakan lain yang membahayakan keamanan sistem. Proses ini merupakan langkah awal yang digunakan *hacker* untuk melakukan intrusi.

2.4.1 Klasifikasi Probing

Beberapa tipe probing yang sering digunakan *hacker* untuk mendapatkan informasi dari target :

1. Port Scan

   Jenis yang paling umum dari probing jaringan adalah scanning port. Scanning port merupakan metode yang digunakan oleh penyusup untuk menemukan layanan yang sedang berjalan pada mesin target. Penyusup kemudian dapat merencanakan serangan pada setiap layanan yang rentan untuk diserang. Metode yang digunakan adalah terhubung ke beberapa port sehingga dapat diketahui port yang merespon atau tidak.

   Cara lain dengan menggunakan half-open SYN scan. Pada scan ini, port scanner terhubung pada beberapa port tetapi menutup koneksi tepat sebelum koneksi secara sempurna terjadi. Karena koneksi sempurna tidak pernah terjadi, sistem operasi dari targer biasanya tidak mencatat scan tersebut.

2. Port Sweeps

   Penyusup mengirim satu set paket ICMP ECHO ke jaringan dan melihat yang memberi respon. Inti dari metode ini adalah mengetahui mesin yang hidup dan tidak. Ketika penyusup mengetahui mesin yang hidup maka penyusup tersebut dapat fokus pada satu mesin untuk diserang. Namun terdapat alasan yang masuk akal untuk melakukan

ping sweeps pada jaringan, administrator jaringan mencoba untuk mengetahui mesin yang hidup dalam jaringan.

3. TCP/IP Stack Fingerprinting

Fingerprinting mengkategorikan sistem operasi berdasarkan nilai unik pada paket yang telah dikirim. Setiap sistem operasi memiliki nilai yang berbeda pada beberapa *field* seperti TTL, ukutan *window* TCP dan TCP option. Selain itu juga terdapat beberapa *field* yang dapat diperiksa seperti nilai dari *Type of Service* (ToS) dan DF flag.

2.4.2 Aplikasi Probing OS *Fingerprinting*

1. Nmap

Nmap (*Network Mapper*) merupakan aplikasi untuk mengetahui *host* dan layanan pada jaringan komputer serta memetakan jaringan. Untuk mencapai tujuan tersebut, Nmap mengirim paket tertentu ke target dan menganalisa respon nya.

Dengan banyaknya dukungan dari komunitas, Nmap dapat memperluas kemampuan dari mengetahui *host* yang hidup dan *port* yang terbuka hingga dapat menentukan sistem operasi, nama dan versi, waktu hidup, tipe peralatan dan keberadaan *firewall*.

Nmap sendiri dapt digunakan di lingkungan Linux, Microsoft Windows, Solaris, HP-UX, BSD, AmigaOS dan SGI IRIX.

2. Xprobe2

Selain Nmap, aplikasi OS *fingerprinting* aktif yang sering digunakan di lingkungan linux adalah Xprobe2. Metode yang digunakan untuk mengidentifikasi sistem operasi adalah matriks berdasarkan pendekatan *fingerprinting* yang juga dikenal dengan sebutan pencocokan *fuzzy*.

Tidak seperti aplikasi lainnya, Xprobe2 tidak menjalankan *scan* port pada target. Xprobe2 membutuhkan setidaknya satu port UDP yang tertutup untuk dapat bekerja. Selain menggunakan protokol

UDP, aplikasi ini juga menganalisa hasil *scanning* dari paket ICMP yang dikirim.

3. NetScanTools Pro

Tidak semua aplikasi yang digunakan untuk penelitian ini berjalan pada lingkungan linux. NetScanTools Pro merupakan aplikasi yang bertujuan mengumpulkan informasi yang berjalan di lingkungan windows. NetScanTools Pro juga digunakan oleh ahli jaringan profesional untuk melakukan *troubleshooting* pada jaringan komputer.

Beberapa keunggulan NetScanTools Pro adalah penghematan waktu yang dibutuhkan untuk mengumpulkan informasi pada jaringan komputer, menghasilkan hasil yang jelas pada antarmuka dan tersedia dalam USB *flash drive*. Namun NetScanTools Pro bukan aplikasi gratis sehingga pengguna harus membeli pada vendor atau menggunakan versi demo yang dapat berjalan selama satu bulan.

4. Autoscan Network

Autoscan Network merupakan aplikasi pengamat jaringan yang tujuan utamanya adalah menampilkan daftar perangkat yang terhubung di jaringan. Autoscan Network dapat diunduh secara bebas di internet dan dapat berjalan pada berbagai lingkungan sistem operasi seperti linux, unix, mac dan windows.

5. Jcannon.py

Aplikasi sederhana yang ditulis dalam bahasa pemrograman python. Jcannon.py dapat digunakan untuk mengetahui host yang hidup, port yang terbuka dan sistem operasi yang digunakan. Jcannon.py berjalan pada lingkungan linux.

2.4.3 Pola Probing OS *Fingerprinting*

Paket data yang dihasilkan oleh aplikasi probing memiliki perbedaan jika dibandingkan dengan paket normal. Melalui studi pada

situs resmi aplikasi probing, jurnal ilmiah dan pengamatan langsung, penulis dapat mengetahui paket yang dihasilkan oleh aplikasi probing. Dengan mengamati paket-paket tersebut, pola serangan probing dapat diketahui. Pola serangan tersebut dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1 Pola Probing OS *Fingerprinting*

(Sumber : [LYO-07] dan [RYA-03])

Protokol	TCP, UDP, ICMP, SNMP, DNS
Length	60, 66, 70, 72, 73, 74, 85, 118, 162, 192, 342
Flag	SYN, SYN ECN CWR, FIN SYN PSH URG, ACK, FIN PSH URG, None
Sequence	0, 1, 295, 296, 1000, 62809, 500000000 <
Info	TCP : Tsval = 4294967295 WS = 1024 Window = 8190 ICMP : Timestamp request Address mask request Information request SNMP : get-request 1.3.6.1.2.1.1.1.0 DNS : Standard query response 0x9a61 A 205.206.231.10

2.5 Naïve Bayes

2.5.1 Teorema Naïve Bayes

Naïve Bayes Classifier merupakan sebuah metode klasifikasi yang berakar pada teorema Bayes. Ciri utama dari *Naïve Bayes Classifier* ini adalah asumsi yang sangat kuat (naif) akan independensi dari masing-masing kondisi/kejadian. Sebelum menjelaskan *Naïve Bayes Classifier* ini, akan dijelaskan terlebih dahulu Teorema Bayes yang menjadi dasar dari metoda tersebut.

Pada teorema Bayes, bila terdapat dua kejadian yangterpisah (misalkan A dan B), maka teorema Bayes dirumuskan sebagai berikut:

$$P(A|B) = \frac{P(A)}{P(B)} P(B|A) \quad \text{.....(pers. 1)}$$

Teorema Bayes sering pula dikembangkan mengingat berlakunya hukum probabilitas total, menjadi sepertiberikut:

$$P(A|B) = \frac{P(A)P(B|A)}{\sum_{i=1}^{n} P(Ai|B)} \quad \text{.....(pers. 2)}$$

dimana $A_1 U\, A_2\, U \ldots U\, A_n = S$

Untuk menjelaskan teorema *Naïve Bayes*, perlu diketahui bahwa proses klasifikasi memerlukan sejumlah petunjuk untuk menentukan kelas apa yang cocok bagi sampel yang dianalisis tersebut. Karena itu, teoremaBayes di atas disesuaikan sebagai berikut:

$$P(C|F_1, \ldots, F_n) = \frac{P(C)P(F_1,\ldots,F_n|C)}{P(F_1,\ldots,F_n)} \quad \text{.....(pers. 3)}$$

Dimana variabel C merepresentasikan kelas, sementara variabel F_1 ... F_n merepresentasikan karakteristik-karakteristikpetunjuk yang dibutuhkan untuk melakukan klasifikasi. Maka rumus tersebut menjelaskan bahwa peluang masuknya sampel dengan karakteristik tertentu dalam kelas C (*posterior*) adalah peluang munculnya kelas C (sebelum masuknya sampel tersebut, seringkali disebut *prior*), dikali dengan peluang kemunculan karakteristik-karakteristik sampel pada kelas C (disebut juga *likelihood*), dibagi dengan peluang kemunculan karakteristik-karakteristik sampel secara global (disebutjuga *evidence*). Karena itu, rumus sebelumnya dapat pula ditulis secara sederhana sebagai berikut:

$$Posterior = \frac{prior\ \ x\ likeli\ hood}{evidence} \quad \text{.....(pers. 4)}$$

Nilai *evidence* selalu tetap untuk setiap kelas pada satu sampel. Nilai dari *Posterior* tersebut yang nantinya akan dibandingkan dengan nilai-nilai *Posterior* kelas menentukan ke kelas apa suatu sampel akan diklasifikasikan.

Penjabaran lebih lanjut rumus Bayes tersebut dilakukan dengan menjabarkan $P(F_1,\ldots,F_n|C)$ menggunakan aturan perkalian, menjadi sebagai berikut:

$$P(F_1, \dots, F_n|C) = P(F_1|C)\, P(F_2, \dots, F_n|C, F_1)$$
$$P(F_1, \dots, F_n|C) = P(F_1|C)\, P(F_2|C, F_1)\, P(F_3, \dots, F_n|C, F_1, F_2)$$
$$P(F_1, \dots, F_n|C) = P(F_1|C)\, P(F_2|C, F_1) \dots P(F_n|C, F_1, F_2, \dots, F_{n-1})$$

.....(pers. 5)

Dapat dilihat bahwa hasil penjabaran tersebut menyebabkan semakin banyak dan semakin kompleksnya faktor-faktor syarat yang mempengaruhi nilai probabilitas, yang hampir mustahil untuk dianalisa satu-persatu.Akibatnya, perhitungan tersebut menjadi sulit untukdilakukan.

Di sinilah digunakan asumsi independensi yang sangat tinggi (naif), bahwa masing-masing petunjuk *(F_1, F_2 ... F_n)*saling bebas (independen) satu sama lain. Dengan asumsi tersebut, maka berlaku suatu kesamaan sebagai berikut:

$$P(F_i|F_j) = \frac{P(F_i \cap F_j)}{P(F_j)} = \frac{P(F_i)P(F_j)}{P(F_j)} = P(F_i)$$ .....(pers. 6)

untuk i $\neq$ j, sehingga

$$P(F_i|C, F_j) = P(F_i|C)$$ .....(pers. 7)

Dari persamaan di atas dapat disimpulkan bahwa asumsi independensi naif tersebut membuat syarat peluang menjadi sederhana, sehingga perhitungan menjadi mungkin untuk dilakukan. Selanjutnya, penjabaran*P(F_1,...,F_n|C)* dapat disederhanakan menjadi sepertiberikut:

$$P(F_1 \dots F_n|C) = P(F_1|C)\, P(F_2|C) \dots P(F_n|C)$$
$$P(F_1 \dots F_n|C) = \prod_{i=1}^{n} P(F_i|C)$$ .....(pers. 8)

Dengan kesamaan di atas, persamaan teorema Bayesdapat dituliskan sebagai berikut:

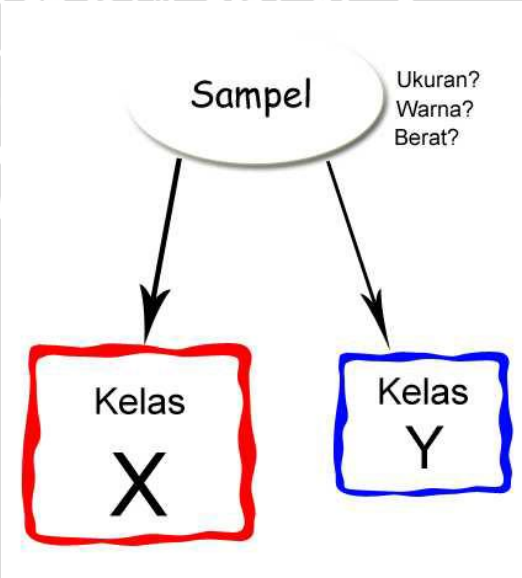
$$P(C|F_1 \dots F_n) = \frac{1}{P(F_1, F_2 \dots F_n)} P(C) \prod_{i=1}^{n} P(F_i|C)$$

$$P(C|F_1 \dots F_n) = \frac{P(C)}{Z} \prod_{i=1}^{n} P(F_i|C)$$ .....(pers. 9)

Persamaan di atas merupakan model dari teorema *Naïve Bayes* yang selanjutnya akan digunakan dalam proses klasifikasi dokumen. Adapun Z merepresentasikan *evidence* yang nilainya konstan untuk semua kelas pada satu sampel.

2.5.2 Klasifikasi Naïve Bayes

Menurut Agus Mulyanto, klasifikasi adalah proses untuk menemukan modelatau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu obyek. Oleh karena itu, kelas yang ada tentulah lebih dari satu. Penentuan kelas dari suatu dokumen dilakukan dengancara membandingkan nilai probabilitas suatu sampelberada di kelas yang satu dengan nilai probabilitas suatu sampel berada di kelas yang lain.



Gambar 2.5 Ilustrasi Proses Klasifikasi

Dengan persamaan teorema *Naïve Bayes*, terdapatkan nilai $P(C|F_1...F_n)$, yaitu nilai peluang suatu sampel dengan karakteristik $F_1...F_n$ berada dalam kelas C, atau dikenal dengan istilah *Posterior*. Umumnya kelas yang ada tidak hanya satu, melainkan lebih dari satu.

Sebagai contoh, ahli statistik ingin mengklasifikasikan sampel kucing ke dalam jenis kelaminnya. Oleh karena itu, terdapat dua kelas yaitu jantan

dan betina. Suatu sampel kucing akan diklasifikasikan ke dalam satu kelas saja, entah itu jantan atau betina, dengan melihat petunjuk-petunjuk yang ada (misalnya berat badan,panjang ekor, dll).

Penentuan kelas yang cocok bagi suatu sampel dilakukan dengan cara membandingkan nilai *Posterior* untuk masing-masing kelas, dan mengambil kelas dengan nilai *Posterior* yang tinggi. Secara matematis klasifikasi dirumuskan sebagai berikut:

$$C_{NB} = argmax_{c \in C} P(c) \prod_{i=1}^{n} P(fi|c) \quad \text{.....(pers. 10)}$$

dengan c yaitu variabel kelas yang tergabung dalam suatu himpunan kelas C. Dapat dilihat bahwa rumusan di atas tidak memuat nilai *Evidence* (Z). Hal ini disebabkan karena *evidence* memiliki nilai yang positif dan tetap untuk semua kelas sehingga tidak mempengaruhi perbandingan nilai *Posterior*. Karena itu, faktor Z ini dapat dihilangkan. Perlu menjadi perhatian pula bahwa metoda *Naïve Bayes classifier* ini dapat digunakan bila sebelumnya telah tersedia data yang dijadikan acuan untuk melakukan klasifikasi.

Sebagai contoh, terdapat dua kelompok merek sepatu (X dan Y), dimana terdapat 3 petunjuk yang digunakan misalnya warna sepatu (merah, hitam), bahan sepatu (kulit, sintetis) dan model sepatu (Tali, Velkro). Data lengkap terdapat pada tabel 2.2.

Tabel 2.2 Contoh Data Klasifikasi Naïve Bayes

Warna	Bahan	Model	Jenis
Merah	Kulit	Tali	X
Hitam	Kulit	Tali	X
Merah	Sintetis	Velkro	Y
Hitam	Kulit	Velkro	Y
Hitam	Sintesis	Tali	X
Hitam	Sintesi	Velkro	Y

Bila terdapat sampel sepatu Hitam, Sintetis, Tali (tidak ada pada data di atas), klasifikasi dapat dilakukan dengan menggunakan *Naïve Bayes classifier*. Pertama-tama harus dicari terlebih dahulu *Posterior* X dan Y untuk sampel tersebut seperti paga gambar 2.6 dibawah ini.

$P(X) = 3/6 = 0.5$	$P(Y) = 0.5$
$P(Hitam\|X) = 2/3 = 0.66$	$P(Hitam\|Y)=1/3=0.33$
$P(Sintetis\|X)=1/3=0.33$	$P(Sintetis\|Y)=2/3=0.66$
$P(Tali\|X)=2/3=0.66$	$P(Tali\|Y)=1/3=0.33$

$Posterior\ X = P(X)\ P(Hitam|X)\ P(Sintetis|X)\ P(Tali|X)$

$= 0.5 \times 0.66 \times 0.33 \times 0.66 = 0.072$

$Posterior\ Y = P(Y)\ P(Hitam|Y)\ P(Sintetis|Y)\ P(Tali|Y)$

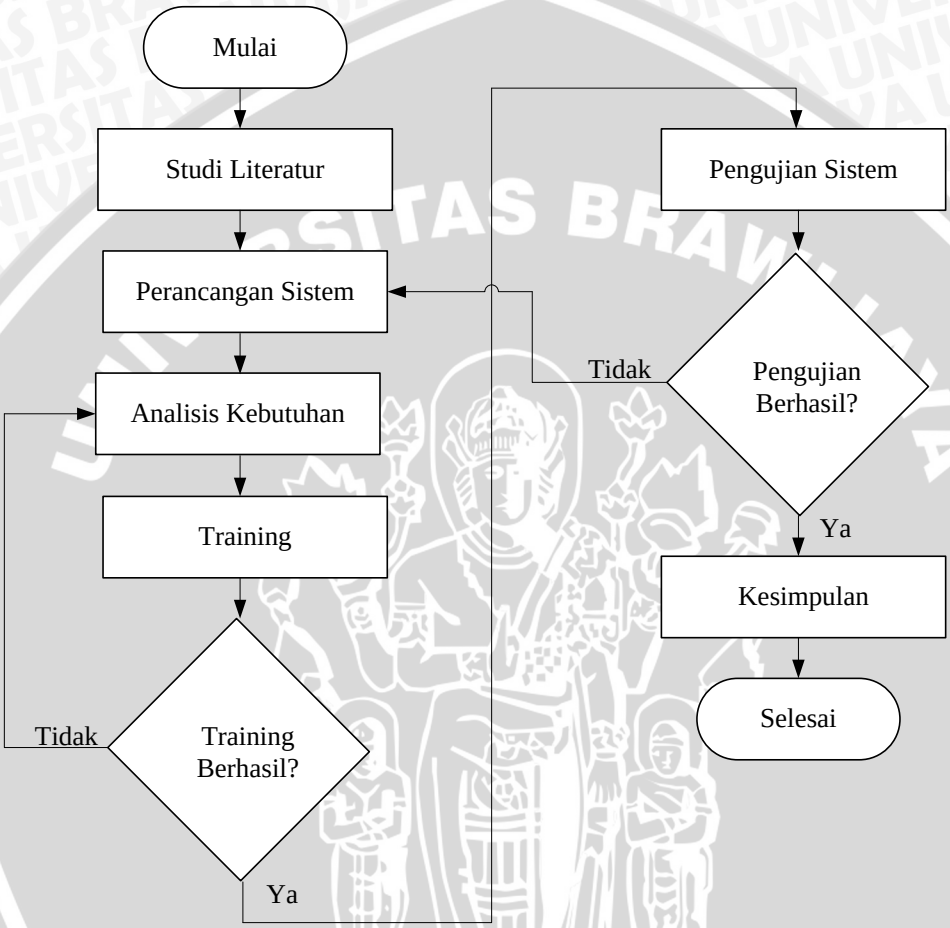
$= 0.5 \times 0.33 \times 0.66 \times 0.33 = 0.034$

Gambar 2.6 Pencarian Posterior X dan Posterior Y

Merujuk pada gambar 2.6, setelah dilakukan proses klasifikasi didapatkan hasil posterior X 0.072 dan posterior Y 0.034. Karena *Posterior X* bernilai lebih besar dari *Posterior Y*, maka sampel sepatu tersebut bermerek X.

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Pada bab ini dijelaskan metodologi penelitian, perancangan, dan analisis kebutuhan dari sistem yang akan dibuat. Metodelogi penelitian yang akan dilakukan pada skripsi ini secara umum dapat dilihat pada diagram alir berikut.



Gambar 3.1 Diagram Alir Keseluruhan Pelaksanaan Penelitian

3.1. Studi Literatur

Studi literatur dilakukan untuk menambah referensi dan pengetahuan yang diperlukan dalam mengerjakan penelitian dan penulisan laporan penelitian. Adapun yang perlu masuk dalam bahan studi literatur adalah dasar-dasar teori untuk dapat merancang dan membangun *anomaly-based intrusion detection system* dan cara pengujian sistem yang meliputi :

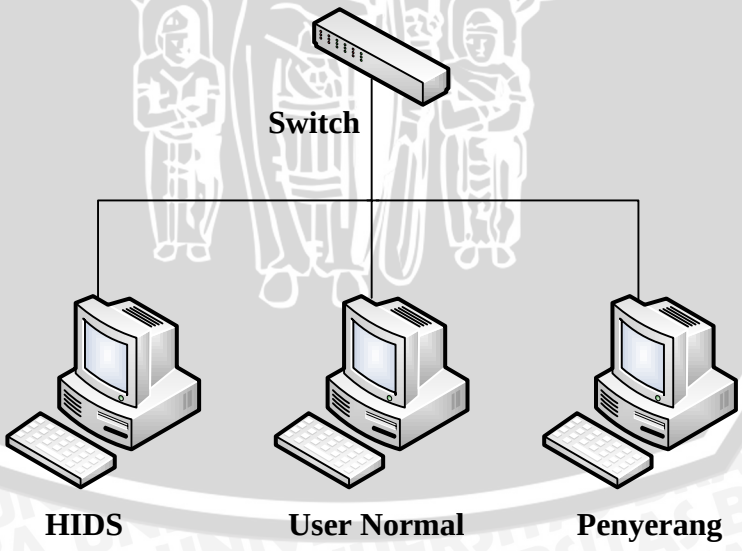
- *Intrusion Detecton System*
- Probing OS *Fingerprinting*
- Wireshark/Tshark
- Python
- Klasifikasi Naïve Bayes

3.2. Perancangan Sistem

Perancangan sistem memberikan gambaran mengenai sistem yang akan dibuat dalam penelitian yang meliputi sistem perangkat keras dan perangkat lunak. Perancangan sistem bertujuan sebagai acuan dalam implementasi sistem dan untuk melakukan analisis kebutuhan yang akan dipergunakan dalam penelitian.

3.2.1 Topologi Jaringan

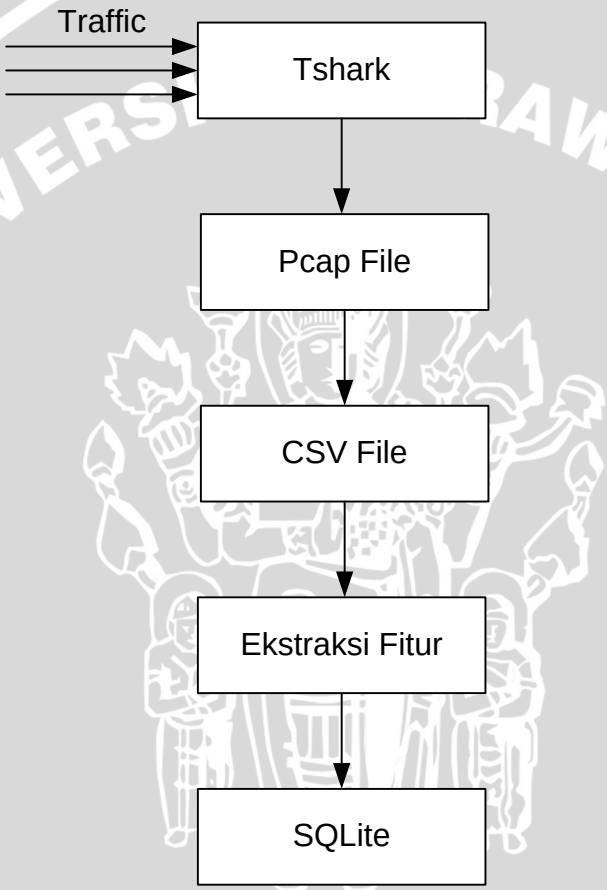
Penelitian dilaksanakan pada jaringan LAN yang terdiri dari switch dan minimal 3 *host*. *Host* tersebut terdiri dari satu komputer target, satu pengguna normal, dan satu penyerang. Topologi sistem secara umum digambarkan sebagai berikut :



Gambar 3.2 Topologi Sistem

3.2.2 Pengambilan Data

Pengambilan data dilakukan dengan cara menangkap paket yang masuk dari setiap *host* menggunakan Tshark. Paket yang ditangkap oleh Tshark akan ditulis dalam sebuah *file* pcap. *File* pcap tersebut merupakan file *binary* dari Tshark, tidak dapat dibaca langsung oleh aplikasi lain sehingga harus dikonversi menjadi file CSV. Bagan pengambilan data hingga tahap *preprocessing* dapat dilihat pada gambar 3.3.



Gambar 3.3 Bagan Pengambilan Data Uji

Hasil dari konversi tersebut adalah *file coma separated value* (CSV) yang dapat dibaca oleh aplikasi pembaca text umum yang terdapat secara default pada sistem opreasi. Sebelum data dimasukkan ke dalam database, perlu adanya ekstraksi fitur yang berfungsi mengambil fitur yang dibutuhkan oleh sistem. Fitur yang diperlukan antara lain source IP Address,

protocol, length, flag, sequence, dan info. Selengkapnya dapat dilihat pada tabel 3.1 berikut

Tabel 3.1 Fitur IDS

No	Nama Fitur	Tipe
1	IP Address	String
2	Protocol	String
3	Length	Integer
4	Flag	String
5	Sequence	Integer
6	Info	String

Fitur tersebut dimasukkan dalam database SQLite yang berisikan atribut sebagai berikut:

Tabel 3.2 Atribut Pada Database

No	Nama Atribut	Tipe	Keterangan
1	No	integer	Nomor urut data
2	IP Address	String	Alamat asal paket data
3	Protocol	String	Protocol yang digunakan
4	Length	Integer	Length paket
5	Flag	String	Flag yang digunakan
6	Sequence	Integer	Nilai sequence paket
7	Info	String	Info tambahan pada paket
8	Label	String	Nilai dari hasil klasifikasi. 0 merupakan paket normal dan 1 merupakan srangan

3.3. Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendata kebutuhan sistem *anomaly-based host intrusion detection system*. Pada tahap ini dilakukan identifikasi kebutuhan perangkat lunak dan perangkat keras dari sistem yang dirancang.

Dengan demikian diharapkan dapat mempermudah dalam mendesain sistem dan hasil analisis terhadap sistem yang dibuat dapat lebih akurat.

Perangkat keras yang dibutuhkan adalah tiga buah komputer sebagai host. Host penyerang, host pengguna normal dan satu target yang telah terinstal *anomaly based intrusion detection system*. Perangkat lunak yang digunakan adalah sistem operasi Ubuntu 13.04 pada target sedangkan pada host menggunakan Backtrack 5 R3 dan Microsoft Windows. Kebutuhan perangkat keras dan perangkat lunak dapat dilihat pada tabel 3.3 berikut:

Tabel 3.3 Kebutuhan Hardware dan Software

Perangkat keras / Perangkat lunak	Pengguna Normal dan Penyerang	Target
CPU	Processor Intel Core i3	Processor Intel Core i3
Memory	2 GB RAM	2 GB
Disk Space	40 GB	60 GB
Sistem Operasi	Backtrack 5 R3 Microsoft Windows	Ubuntu 13.04
Aplikasi	Nmap 6.01 Xprobe2 2.1-bt2 NetScanTools Pro 11.32.r1 Autoscan Network 1.5 jcannon.py Google Chrome Ping Putty Samba sinFP3 1.21	Wireshark/Tshark SQLite IDS

3.4. Training

Penulis menyediakan 3 jenis dataset training pada penelitian IDS ini. Setiap data telah diketahui termasuk pada kelas probing atau normal. Dataset pertama terdiri dari 32 paket probing dan 68 paket normal, dataset kedua terdiri dari 32

paket probing dan 168 paket normal, sedangkan dataset ketiga terdiri dari 32 paket probing dan 268 paket normal.

3.5. Pengujian Sistem

Secara garis besar pengujian dibagi menjadi dua bagian, simulasi dan *real attack*. Pada pengujian simulai, penulis menyediakan data uji dalam jumlah kecil yang telah diketahui paket probing atau normal. Sedangkan pada pengujian *real attack*, penulis membuat jaringan yang mengacu pada gambar 3.2 diatas. Data uji didapat langsung dari paket yang dikirimkan dari setiap host pada jaringan tersebut.

3.5.1. Pengujian Simulasi

Penulis menggunakan 3 buah skenario pengujian untuk mengukur tingkat akurasi dari*anomaly based intrusion detection system*. Selain itu, pada pengujian ini dapat dilihat data latih yang menghasilkan tingkat akurasi terbaik. Pada setiap skenario, sistem menggunakan data latih untuk mengklasifikasi 3 tipe data uji yang berbeda jumlah data probing dan data normal. Data lengkap skenario terdapat di tabel 3.4.

Tabel 3.4 Skenario Pengujian Simulasi

	Data Latih	**Data Uji**
Skenario 1	100 data latih 32 probing dan 68 normal	10 probing dan 20 normal
		20 probing dan 40 normal
		30 probing dan 60 normal
Skenario 2	200 data latih 32 probing dan 168 normal	10 probing dan 20 normal
		20 probing dan 40 normal
		30 probing dan 60 normal
Skenario 3	300 data latih 32 probing dan 268 normal	10 probing dan 20 normal
		20 probing dan 40 normal
		30 probing dan 60 normal

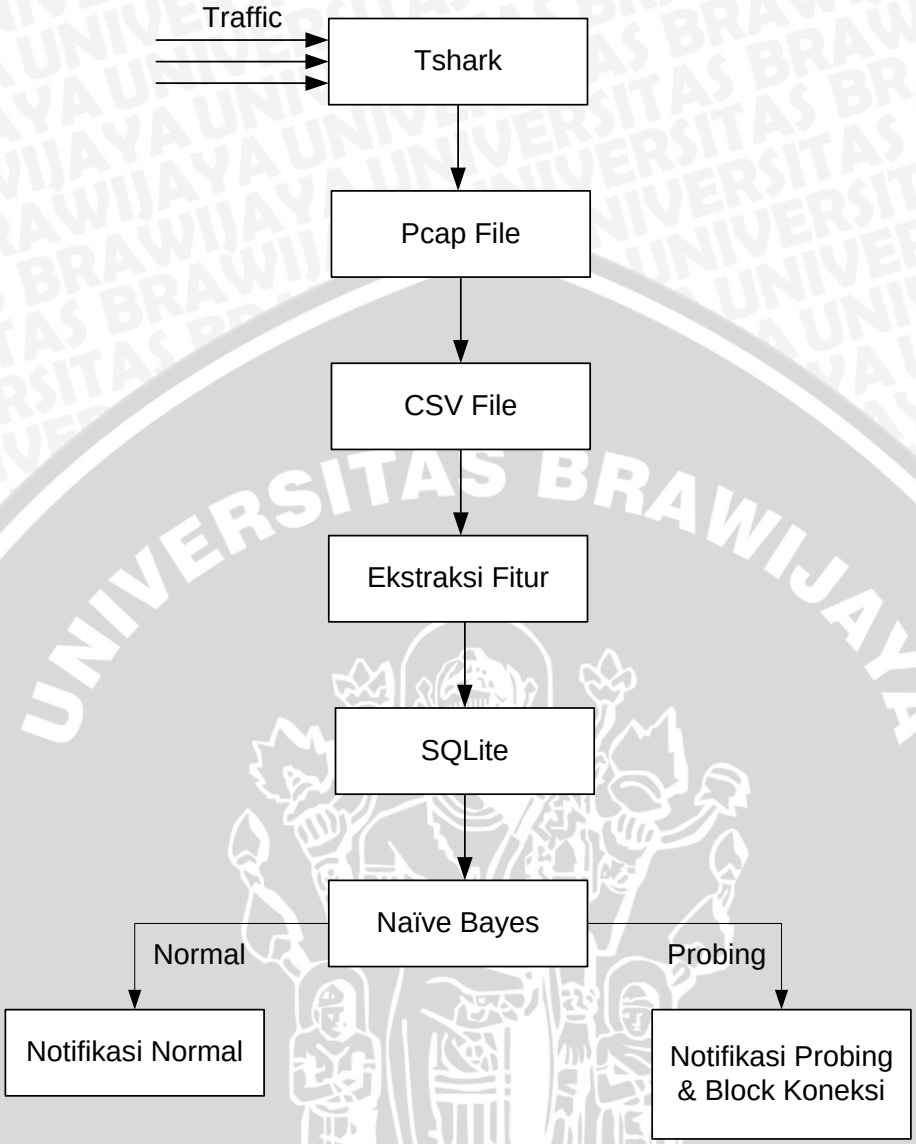
3.5.2. Pengujian *Real Attack*

Pada pengujian ini terdapat satu kompoter target yang berfungsi menangkap dan mengklasifikasi paket, satu *host* yang berfungsi mengirimkan paket normal dan satu *host* yang mengirimkan paket probing maupun normal. Sistem operasi pada *host* penyerang menggunakan Backtrak 5 R3. Selain menggunakan aplikasi probing yang telah ada pada sistem operasi tersebut, penulis menambahkan aplikasi probing sinFP3 untuk menguji apakah sistem dapat mendeteksi serangan dari aplikasi lain. Setiap skenario pengujian dilakukan selama 300 detik (5 menit). Data latih yang digunakan pada pengujian *real attack* adalah data latih yang menghasilkan *recall,* presisi dan akurasi paling tinggi pada pengujian simulasi. Berikut data skenario pada pengujian ini:

Tabel 3.5 Pengujian *Real Attack*

	OS Host Penyerang	**Aplikasi**
Pengujian *Real Attack*	Backtrack Linux 5 R3	Probing: • Nmap • Xprobe2 • jcannon.py • sinFP3 Non Probing: • Google Chrome • Ping • Putty • Samba

Dari ekstraksi file CSV sebelumnya, database telah menyimpan fitur yang dibutuhkan oleh sistem. Klasifikasi oleh aplikasi IDS akan dilakukan pada setiap data yang terdapat pada database. Jika data dianggap sebuah probing, maka akses alamat IP dari paket tersebut akan ditutup. Bagan pengujian dapat dilihat pada gambar 3.4.



Gambar 3.4 Bagan Klasifikasi Paket Data

3.6. Kesimpulan dan Saran

Kesimpulan diambil setelah perancangan, implementasi, pengujian dan analisis selesai dilakukan. Kesimpulan disusun berdasarkan hasil pengujian dan analisis sistem yang telah dibuat. Isi dari kesimpulan diharapkan dapat menjadi acuan pada penelitian lain untuk mengembangkan sistem pendeteksi serangan. Selain itu, pada akhir penulisan terdapat saran yang bertujuan untuk penyempurnaan penelitian ini.

BAB IV
IMPLEMENTASI

Pada bab implementasi dibahas mengenai langkah-langkah dalam pembuatan *anomaly based intrusion detection system*. Langkah penerapan mengacu pada tahapan perancangan yang terdiri dari sebuah komputer sebagai target serangandan dua buah kompoter *host* yang berfungsi untuk mengirimkan paket normal dan paket probing OS *fingerprinting* secara bergantian. Implementasi meliputi lingkungan perangkat keras dan implementasi perangkat lunak.

4.1 Lingkungan Implementasi

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pembentukan IDS meliputi :

1. Komputer target
   - Processor Intel Core i3 CPU
   - 2 GB RAM
   - 465 GB harddisk Drive
2. Komputer pengguna dan penyerang
   - Processor Intel Core i3 CPU
   - 2 GB RAM
   - 465 GB harddisk Drive
3. Peralatan Jaringan Komputer
   - Kabel UTP Cat 5
   - Konektor RJ-45
   - Swicth 16 Port

4.1.2 Lingkungan Perangkat Lunak

1. Komputer target
   - Sistem Operasi Ubuntu 13.04
   - Wireshark/Tshark Packet Sniffer 1.9.2
   - Bash Script
   - Pyhton 3

- SQLite 3

2. Komputer pengguna dan penyerang
   - Sistem Operasi Backtrack Linux 5 R3 / Windows 7
   - Nmap Network Mapping 6.01
   - Xprobe2 2.1 bt-2
   - NetScanTools Pro 11.32.r1
   - Autoscan Network 1.5
   - Jcannon.py

4.2 Proses Pembentukan Sistem

Berdasarkan hasil perancangan pada bab 3 tentang analisa proses pembentukan dataset dalam sistem pendektesian probing OS *fingerprinting*, pada bab ini akan dijelaskan implementasi dari perancangan yang telah dibuat tersebut.

4.2.1 Pembuatan Dataset

Dalam pembuatan dataset dan klasifikasi, penulis menggunakan bash shell agar sistem lebih mudah dijalankan. Di dalam file tersebut terdapat perintah untuk menjalankan Tshark yang menangkap paket pada *interface* eth0 dengan tujuan dari paket adalah IP target dan disimpan dalam keluaran file *.pcap. Pseudocode dapat dilihat pada gambar 4.1 dan kode program dapat dilihat pada kode program 4.1

Pseudocode Menangkap Paket Data

```
variable string FileName
if FileName = not null then
     print "Menangkap paket dan disimpan di file 'Filename'"
     get IPAddress
     sniff packet
     save packet to 'FileName'
else
     print "Masukkan nama file. Syntax :"
     print "./1_capture.sh -o 'namafile.pcap'"
end if
```

Gambar 4.1 Pseudocode Menangkap Paket Data

1_capture.sh

```
#!/bin/bash

if [ -n "$2" ]; then
      echo "################################################"
      echo "Menangkap paket dan disimpan di file $2"
      echo "          Tekan Ctrl+Z untuk berhenti          "
      echo "################################################"

      ip="$(ifconfig | grep -A 1 'eth0' | tail -1 | cut -d
':' -f 2 | cut -d ' ' -f 1)"
         /home/amrisinclair/Wireshark/tshark -i eth0 dst $ip
-w $2

elif [ -z "$2"]; then
         echo "Masukkan nama file. Syntax :"
      echo "./1_capture.sh -o 'namafile.pcap'"
fi
```

Kode Program 4.1 File Bash Menangkap Paket

Syntax Tshark

```
tshark –i eth0 dst $ip –w $2
```

Kode Program 4.2 Syntax Tshark

Penjelasan dari kode program 4.2 adalah sebagai berikut:

- `Tshark` nama aplikasi yang digunakan
- `–i eth0` syntax ini menunjukkan interface yang digunakan untuk menangkap data
- `dst $ip` berfungsi untuk menangkap paket data dengan IP tujuan adalah IP dari target. Nilai dari `$ip` didapat dari baris kode sebelumnya pada kode program 4.1
- `–w $s2` seluruh paket ditulis ke dalam sebuah file yang sesuai dengan value dari `$s2`. Nilai dari `$s2` didapat dari argumen ketika pengguna menjalankan file bash.

Untuk mendapatkan data paket probing OS fingerprinting, penulis menggunakan 5 aplikasi diantaranya Nmap, Xprobre2, dan jcannon.py yang digunakan di lingkungan Linux sedangkan Autoscan Network dan NetScan Tools yang berjalan di lingkungan Windows. Aplikasi yang berjalan di lingkungan linux dijalankan dalam mode *text based* sedangkan pada lingkungan windows menggunakan antarmuka grafis. Syntax yang digunakan untuk menjalankan aplikasi probing di lingkungan Linux dapat dilihat pada kode program 4.3.

Bash uji	
Nmap	`nmap –O <ip-target>`
Xprobe2	`xprobe2 <ip-target>`
Jcannon.py	`./jcannon.py <ip-target> oscan`

Kode Program 4.3 Syntax Aplikasi Probing Linux

4.2.2 Proses Preprocessing

File yang dihasilkan oleh Tshark masih berupa format binary yang tidak bisa dibaca langsung oleh pengguna sehingga harus dikonversi terlebih dahulu ke file ASCII. Hal ini dilakukan agar file menjadi *human readable*. Konversi file Wireshark menjadi file ASCII juga menggunakan apliaksi yang sama, Tshark. Syntax dapat dilihat pada kode program 4.4.

Konversi file pcap ke CSV
`tshark -T fields -n -r $2 -E separator=, -E quote=d -e frame.number -e frame.time_delta -e ip.src -e ip.dst -e ip.proto -e frame.len -e col.Info >uji.csv`

Kode Program 4.4 Konversi pcap ke CSV

Penjelasan syntax pada kode program 4.4

- `Tshark` nama aplikasi yang digunakan
- `-T fields -n –r $2` membaca file dengan value yang berada pada argumen `$2`. Nilai argumen tersebut didapat saat pengguna menjalankan file bash.

- `-E separator=, -E quote=d` memisahkan fitur dengan tanda koma (,) dan (“”)
- `-e frame.number -e frame.time_delta -e ip.src -e ip.dst -e ip.proto -e frame.len -e col.Info` beberapa fitur yang diambil untuk ditulis kembali dalam ASCII
- `>uji.csv` nama file keluaran dari aplikasi tshark

Hasil keluaran konversi

```
"1","0.000000000","8.8.8.8","192.168.247.128","17","129","St
andard query response 0x0e59 No such name"
"2","0.250939000","192.168.247.2","192.168.247.128","17","78
","Standard query response 0xcaf8 "
"3","0.024844000","192.168.247.2","192.168.247.128","17","55
0","Standard query response 0x0e59  A 67.215.65.132"
"4","0.244429000","192.168.247.2","192.168.247.128","17","55
3","Standard query response 0x4028  A 67.215.65.132"
"5","0.704775000","192.168.247.2","192.168.247.128","17","81
","Standard query response 0x805d "
"6","0.084513000","67.215.65.132","192.168.247.128","6","60"
,"80 > 47453 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
MSS=1460"
"7","0.001011000","67.215.65.132","192.168.247.128","6","60"
,"80 > 47453 [ACK] Seq=1 Ack=285 Win=64240 Len=0"
"8","0.037756000","67.215.65.132","192.168.247.128","6","237
","HTTP/1.0 303 See Other "
"9","0.000526000","67.215.65.132","192.168.247.128","6","60"
,"80 > 47453 [ACK] Seq=185 Ack=286 Win=64239 Len=0"
"10","0.131691000","192.168.247.2","192.168.247.128","17","2
10","Standard query response 0x226e  A 67.215.67.9"
```

Gambar 4.2 Contoh Data Raw Konversi Tshark

Dataset pada gambar 4.2 diatas telah dikonversi dalam format ASCII CSV. Terdapat informasi seperti nomor koneksi, waktu koneksi, alamat IP asal, alamat IP tujuan, protokol yang digunakan, alamat port asal, alamat port tujuan, flag, window size, checksum dan beberapa informasi tambahan.

Penulis membuat sebuah program menggunakan bahasa pemrograman python dengan memanfaatkan regular expression untuk melakukan filterisasi informasi yang diperlukan. Informasi yang diambil meliputi alamat IP tujuan, protokol, length, flag dan informasi unik dari setiap paket. Informasi tersebut disortir dan dimasukkkan ke dalam database. Proses preprocessing dan penyimpanan dalam database ditujukkan pada kode program di lampiran 1 dan gambar 4.3.

Pseudocode Preprocessing Data

```
variable string dataIP, string dataPro, integer
dataLength,

string dataFlag, integer dataSeq, string dataInfo
read 'uji.csv'
connect to database

for kolom in 'uji.csv':
     if IPAddress = not null then
          dataIP = IPAddress
     else
          dataIp = '<None>'
     end if

     if Protocol = not null then
          dataPro = Protocol
     else
          dataPro = '<None>'
     end if

     if Length = not null then
          dataLength = length
     else
          datalength = '<None>'
     end if

     if Flag = not null then
          dataFlag = Flag
     else
          dataflag = '<None>'
     end if

     if Sequence = not null then
          dataSeq = Sequence
     else
          dataSeq = '<None>'
     end if

     if Info = not null then
```

```
            dataInfo = '1'
      else
            dataPro = '0'
      end if


      print dataIP, dataPro, dataLength, dataFlag,
dataSeq,

dataInfo
      save to database dataIP, dataPro, dataLength,
dataFlag, dataSeq, dataInfo
```

Gambar 4.3 Pseudocode Preprocessing Data

4.2.3 Klasifikasi Naïve Bayes

Proses ini melakukan klafisikasi terhadap dataset yang telah disediakan. Algoritma yang digunakan adalah naïve bayes. Keseluruhan proses ini ditujukan di kode program pada lampiran 2.

Dalam sekali *running*, program melakukan klasifikasi sebanyak jumlah paket yang terdapat di dalam dataset uji. Jika terdapat 100 data maka dalam sekali eksekusi program tersebut melakukan 100 kali klasifikasi dan menghasilkan keluaran berupa banyaknya serangan yang terdeteksi. Jika sistem mendeteksi sebuah serangan, maka program akan melakukan *update* pada atribut label mejadi bernilai 1 dan jika dianggap paket normal maka akan bernilai 0.

4.2.4 Blok Koneksi

Jika hasil dari klasifikasi terdeteksi probing maka respon dari sistem adalah menutup koneksi pengguna yang mengirimkan serangan. Sistem akan melihat atribut label di database, jika terdapat label dengan nilai 1 maka sistem akan mengambil atribut alamat IP pada baris tersebut dan menjalankan firewall iptables untuk menutup koneksi dari IP asal. Kode program untuk menutup koneksi dapat dilihat pada lampiran 3.

BAB V
PENGUJIAN DAN ANALISIS

Pada bab pengujian dan analisis penulis akan menjabarkan tentang parameter dan hasil pengujian yang telah dilakukan pada sistem. Selain itu juga terdapat analisa hasil klasifikasi dari anomaly based *intrusion detection system*.

5.1 Parameter Pengujian

Merujuk pada gambar 3.4 pada bab metodologi penelitian dan perancangan, sistem deteksi serangan menangkap paket yang dikirimkan pada host target menggunakan aplikasi Tshark. Hasil dari aplikasi tersebut adalah file biner pcap yang harus dikonversi ke dalam file cvs agar bisa dibaca oleh sistem. IDS akan membaca hasil konversi tersebut untuk mendapatkan fitur yang dibutuhkan untuk mendeteksi probing. Fitur tersebut disimpan di dalam database SQLite yang selanjutnya diklasifikasikan menggunakan algoritma naïve bayes. Hasil dari sistem berupa jumlah paket data yang dianggap probing dan normal.

Evaluasi dilaksanakan dengan membandingkan hasil pendeteksian probing OS *fingerprinting* dengan pengamatan langsung terhadap paket data yang ditangkap oleh sistem. Pengujian terhadap sistem dilakukan untuk mengetahui presisi, *recall*, dan akurasi dari sistem yang telah dibangun.

Tingkat presisi, *recall*, dan akurasi dapat dihitung dengan melihat *alert* yang dihasilkan oleh *anomaly based intrusion detection system*. Terdapat empat *alert* yang dihasilkan:

- *True Negative* (TN) : kondisi dimana lalu lintas data berjalan normal dan tidak ada alarm yang dibangkitkan.
- *True Positive* (TP) : kondisi dimana alarm akan ter-*trigger* jika ditemukan kecocokan yang diidentifikasi sebagai serangan.
- *False Negative* (FN) : kondisi dimana tidak ada alarm meski serangan telah masuk ke dalam sistem komputer.
- *False Positive* (FP) : kondisi dimana lalu lintas data berjalan normal namun ada alarm yang dibangkitkan.

Secara umum, keempat *alert* tersebut dapat digambarkan pada tabel 5.1 berikut:

Tabel 5.1 *Alert* IDS

PREDIKSI / AKTUAL	**PROBE**	**NORMAL**
PROBE	TP	FN
NORMAL	FP	TN

Setelah jumlah setiap *alert* telah diketahui, presisi dapat dihitung menggunakan persamaan 11, *recall* menggunakan persamaan 12 dan akurasi menggunakan rumus pada persamaan 13.

Presisi = $\frac{TP}{TP+FP}$ 100%.....(pers. 11)

Recall = $\frac{TP}{TP+FN}$ 100%.....(pers. 12)

Akurasi = $\frac{TP+TN}{TP+TN+FP+FN}$ 100%.....(pers. 13)

5.2 Hasil Pengujian

Sistem hanya dapat memberikan keluaran berupa jumlah dan asal serangan. Untuk itu penulis perlu membandingkan apakah keluaran dari sistem memang benar atau terdapat kesalahan dalam klasifikasi. Dari pengamatan langsung maka didapat presentase jumlah*true negative, true positive, false negative* dan *false positive*. Dengan menerapkan skenario pengujian pada bab metodologi dan perancangan, didapatkan hasil sebagai berikut.

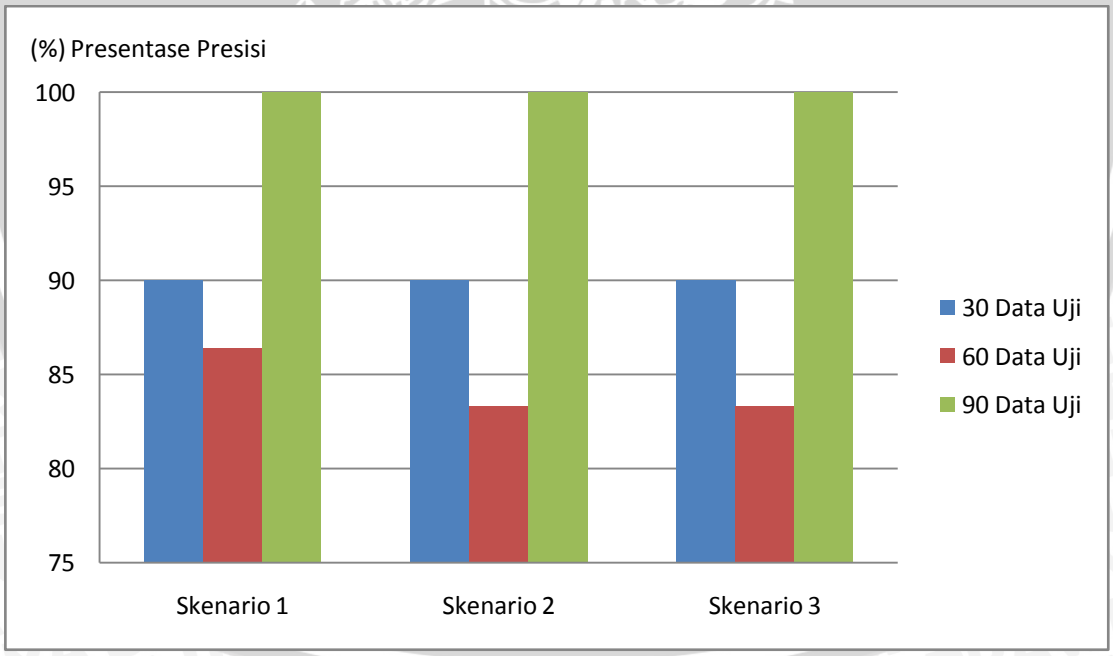
5.2.1. Pengujian Simulasi

Pada pengujian simulasi, tingkat akurasi dan recall paling tinggi didapat pada skenario 1. Sedangkan pada skenario 2 dan skenario 3, tingkat akurasi dan recall mengalami penurunan. Data lengkap pengujian simulasi ini dapat dilihat pada tabel 5.4, gambar 5.1 untuk tingkat presisi, gambar 5.2 untuk tingkat *recall* dan gambar 5.3 untuk tingkat akurasi.

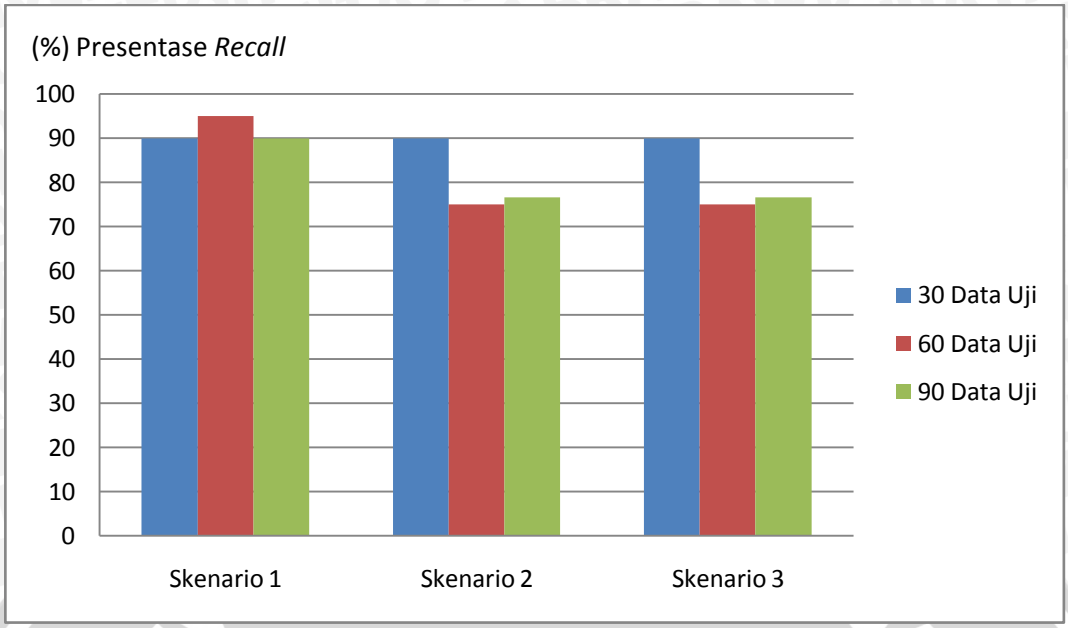
Tabel 5.2 Data Pengujian Simulasi

	Alert	Presisi	Recall	Akurasi
Skenario 1	TN : 19 / 20 = 95 % TP : 9 / 10 = 90 % FN : 1 / 10 = 10 % FP : 1 / 20 = 5 %	90 %	90 %	93.33 %
	TN : 37 / 40 = 97.5 % TP : 19 / 20 = 95 % FN : 1 / 20 = 5 % FP : 3 / 40 = 2.5 %	86.37 %	95 %	93.33 %
	TN : 60 / 60 = 100 % TP : 27 / 30 = 90 % FN : 3 / 30 = 10 % FP : 0 / 60 = 0 %	100 %	90 %	96.67 %
Skenario 2	TN : 19 / 20 = 95 % TP : 9 / 10 = 90 % FN : 1 / 10 = 10 % FP : 1 / 20 = 5 %	90 %	90 %	93.33 %
	TN : 37 / 40 = 92.5 % TP : 15 / 20 = 75 % FN : 5 / 20 = 25 % FP : 3 / 40 = 7.25 %	83.34 %	75%	86.66 %
	TN : 60 / 60 = 100 % TP : 23 / 30 = 90 %	100 %	76.67 %	92.72 %

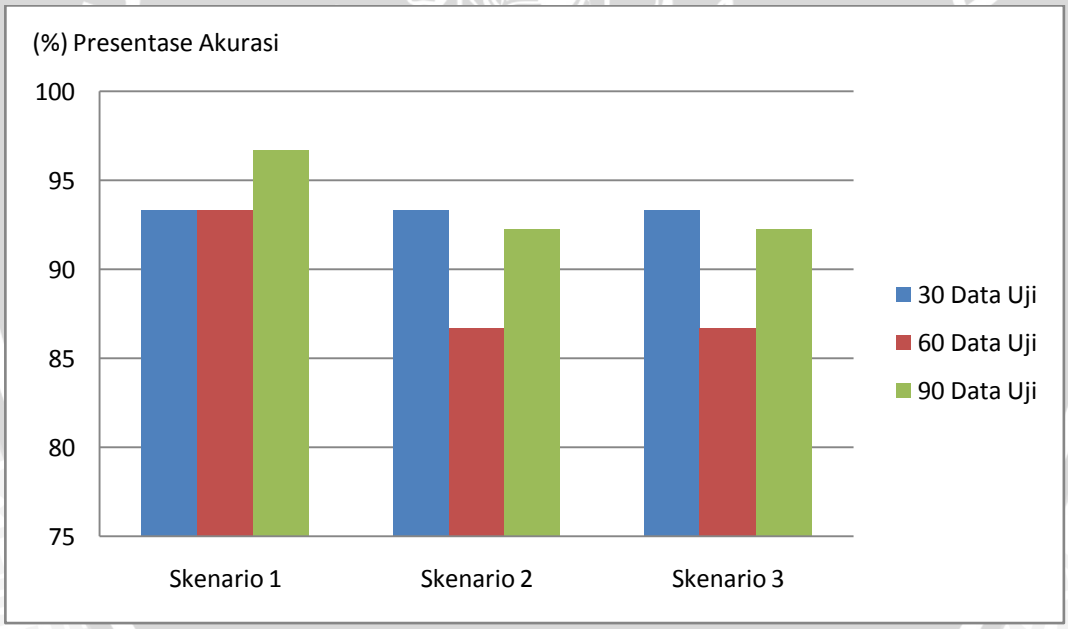
	FN : 7 / 30 = 10 % FP : 0 / 60 = 0 %			
Skenario 3	TN : 19 / 20 = 95 % TP : 9 / 10 = 90 % FN : 1 / 10 = 10 % FP : 1 / 20 = 5 %	90 %	90 %	93.33 %
	TN : 37 / 40 = 92.5 % TP : 15 / 20 = 75 % FN : 5 / 20 = 25 % FP : 3 / 40 = 7.25 %	83.34 %	75 %	86.66 %
	TN : 60 / 60 = 100 % TP : 23 / 30 = 90 % FN : 7 / 30 = 10 % FP : 0 / 60 = 0 %	100 %	76.67 %	92.22 %



Gambar 5.1 Grafik Tingkat Presisi Pengujian Simulasi



Gambar 5.2 Grafik Tingkat *Recall* Pengujian Simulasi



Gambar 5.3 Grafik Tingkat Akurasi Pengujian Simulasi

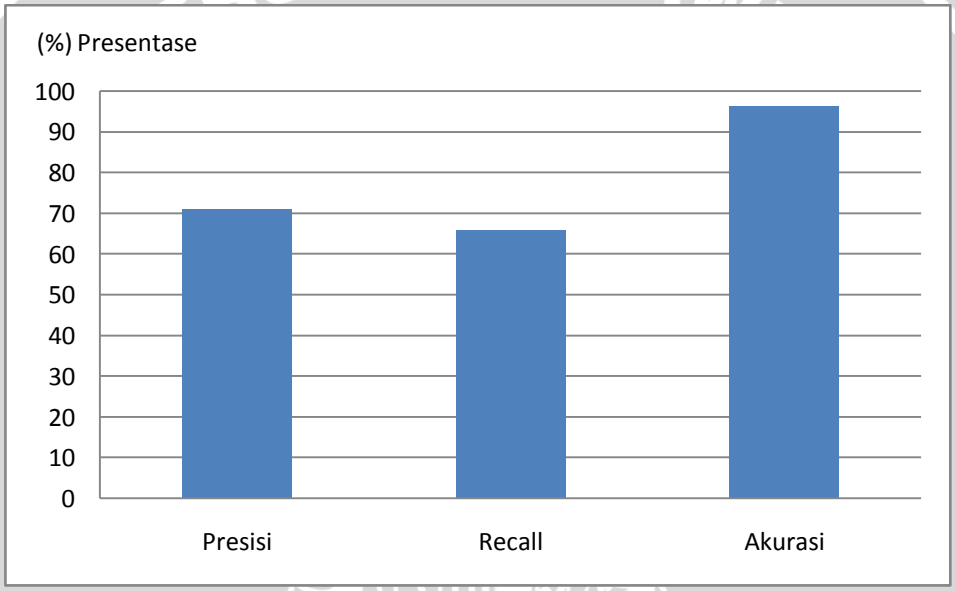
5.2.2. Pengujian *Real Attack*

Mengacu hasil pada pengujian simulasi, pengujian *real attack* menggunakan data latih yang terdiri dari 32 paket probing dan 68 paket normal. Pada pengujian ini, sistem menangkap paket selama 3 menit dari *host* penyerang dan pengguna biasa. Total paket yang ditangkap berjumlah

2492 paket yang terdiri dari 2343 paket normal dan 149 paket probing. Hasil lengkap pengujian *real attack* dapat dilihat pada tabel 5.3 dan gambar 5.4.

Tabel 5.3 Data Pengujian *Real Attack*

	Alert	Presisi	Recall	Akurasi
Real Attack	TN : 2304/2343 = 98.34%	71 %	65.77 %	96.39 %
	TP : 98/149 = 65.77%			
	FN : 51/149 = 34.23%			
	FP : 39/2343 = 1.66%			



Gambar 5.4 Grafik Tingkat Presisi, *Recall* dan Akurasi Pengujian *Real Attack*

5.3 Analisa

Pada pengujian simulasi, baik skenario 1, skenario 2 atau skenario 3 terdapat persamaan. Pada pengujian tersebut, semakin banyak data latih yang digunakan maka tingkat akurasi dan recall dari sistem semakin berkurang. Hal ini disebabkan, penambahan jumlah data latih tidak sebanding. Jumlah data latih probing maksimal berjumlah 33 kombinasi, sedangkan jumlah data latih normal memiliki banyak kombinasi yang banyak ditemukan. Tingkat akurasi dan recall tertinggi dihasilkan pada saat sistem menggunakan data latih dengan jumlah terkecil, 33 probing dan 67 normal.

Variasi data latih tersebut digunakan untuk menjadi data latih yang digunakan sistem pada saat pengujian *real attack*. Dengan menggunakan data latih tersebut tingkat akurasi yang dihasilkan pada saat *real attack* mencapai 96.39%, tingkat presisi yang dihasilkan 78 % dan tingkat *recall* 65.77%. Selain itu, pada pengujian kedua ini ditambahkan satu aplikasi probing yang tidak dimasukkan ke dalam data latih dan sistem dapat mendeteksi serangan tersebut.

BAB VI
PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat penulis ambil dari penelitian yang telah dilakukan dengan berbagai skenario pengujian antara lain:

i) Tingkat presisi, *recall*, dan akurasi *anomaly based intrusion detection system* yang menggunakan algoritma pembelajaran naïve bayes tergantung pada perbandingan jumlah data latih probing dan data latih normal dari setiap kelas.
ii) Pada semua pengujian, klasifikasi paket probing dan normal menggunakan naïve bayes mampu menghasilkan akurasi mencapai 96%.
iii) Sistem pendeteksian serangan menggunakan algoritma pembelajaran naïve bayes dapat mendeteksi serangan yang belum diketahui oleh sistem. Pada penelitian ini, IDS dapat mendeteksi paket probing sinFP3 yang tidak ada di dalam data latih.
iv) Kecepatan sistem dalam melakukan klasifikasi data bergantung pada banyaknya fitur dan perangkat keras yang digunakan.

5.1 Saran

Diperlukan lebih banyak aplikasi *OS fingerprinting* yang harus dipelajari sehingga menambah jumlah kombinasi data latih probing. Aplikasi yang digunakan untuk data latih antara lain Nmap, Xprobe2, Jcannon, Autoscan Network dan NetScanTools Pro. Sedangkan pada pengujian, penulis menambahkan satu aplikasi lagi yaitu sinFP3.

DAFTAR PUSTAKA

[BOL-09] Bo, Li and Dong-Dong, Jian. 2009. *The Research of Intrusion Detection Model Based on Clustering Analysis*. 2009 International Conference on Computer and Communications Security. 24-27.

[GRA-10] Graves, Kimberly. 2010. *Certified Ethical Hacker Study Guide*. Indiana: Wiley Publishing.

[GRE-06] Gregg, Michael. *OSI: Securing the Stack, Layer 4 – Fingerprinting*. http://searchnetworking.techtarget.com/tip/OSI-Securing-the-Stack-Layer-4-Fingerprinting (diakses tanggal 15 Januari 2013)

[LYO-07] Lyon, Gordon. *Remote OS Detection*. http://nmap.org/book/osdetect.html (diakses tanggal 15 Januari 2013)

[NAI-10] Naiping, Song and Genyuan, Zhou. 2010. *A study on Intrusion Detection Based on Data Mining*. 2010 International Conference of Information Science and Management Engineering. 135-138

[PAN-08] Panda, Mrutyunjaya and Patra, Manas Ranja. 2008. *A Comparative Study of Data Mining Algorithms For Network Intrusion Detection*. First International Conference on Emerging Trends in Engineering and Technology.504-507.

[SIM-05] Simson L. Garfinkel. 2005. *Introduction to Computer Security and Privacy*.

[RYA-03] Spangler, Ryan. 2003. *Analysis of Remote Active Operating System Fingerprinting Tools*. Wisconsin : University of Wisconsin.

LAMPIRAN

Lampiran 1. Preprocessing Data

```
import sqlite3 as lite
import csv
import sys
import re

baca =
csv.reader(open('uji.csv','rb'),delimiter=',',quotechar='"')
con = lite.connect('length.db')
cur = con.cursor()

curDEL = con.cursor()
curDEL.execute("delete from uji")

for kolom in baca:
     bacaip =  kolom[2]
     match = re.search(r'\b(?:[0-9]{1,3}\.){3}[0-
9]{1,3}\b',bacaip)
     if match:
          print 'IP Asal              :', match.group()
          dataip = match.group()
     else:
          dataip = '<None>'
          print 'IP Asal              :',dataip

     bacapro = kolom[4]
     match = re.search(r'[\w]+',bacapro)
     if match:
          print 'Protokol Asal        :', match.group()
          datapro = match.group()
     else:
          datapro = '<None>'
          print 'Protokol Asal        :', datapro

     bacalength=kolom[5]
     match = re.search(r'[\d]+',bacalength)
     if match:
          print 'Besar Paket          :', match.group()
          datalength = match.group()
     else:
          datalength = '<None>'
          print 'Besar Paket          :', bacalength

     bacaflag=kolom[6]
     match =
re.search(r'[[]+[A|P|S|F|R|U|E|C|N].+[]]',bacaflag)
     if match:
          dataflag = re.sub(r'\[','',match.group())
          dataflag = re.sub(r',','',dataflag)
```

```
            dataflag = re.sub(r' Reserved','',dataflag)
            dataflag = re.sub(r'\]','',dataflag)
            print "Flag Paket          :",dataflag
      else:
            dataflag = '<None>'
            print "Flag Paket          :",dataflag

      bacaseq=kolom[6]
      match = re.search(r'Seq=[\d]+',bacaseq)
      match2 = re.search(r'seq=[\d]+',bacaseq)
      if match:
            dataseq = re.sub(r'Seq=','',match.group())
            print 'Nilai Sequence      :',dataseq
      elif match2:
            dataseq = re.sub(r'seq=','',match2.group())
            print 'Nilai Sequence      :',dataseq
      else:
            dataseq = '0'
            print 'Nilai Sequence      :',dataseq

      datainfo = '0'
      valueINF = 0
      data17 = '0'
      if datapro == '6':
            bacatsval=kolom[6]
            match = re.search(r'Tsval=4294967295',bacatsval)
            match2 = re.search(r'WS=1024',bacatsval)
            match3 = re.search(r'Seq=1 Win=8190',bacatsval)
            match4 = re.search(r'Seq=0 Win=8190
Urg=0',bacatsval)
            if match:
                  datainfo = '1'
                  valueINF = 1
                  print 'Timestamp Value          :
4294967295'
            elif match2:
                  datainfo = '1'
                  valueINF = 1
                  print 'WS                  : 1024'
            elif match3:
                  datainfo = '1'
                  valueINF = 1
                  print 'Window              : 8190'
            elif match4:
                  datainfo = '1'
                  valueINF = 1
                  print 'Window              : 8190'
      elif datapro == '1':
            bacaicmp=kolom[6]
            match1 = re.search(r'ping',bacaicmp)
            match1b = re.search(r'request',bacaicmp)
            match2 = re.search(r'Address mask
request',bacaicmp)
```

```
            match3 = re.search(r'Timestamp request',bacaicmp)
            match4 = re.search(r'Information
request',bacaicmp)
            if match1 and match1b:
                  if dataseq == '0' or dataseq == '1' or
dataseq == '295' or dataseq == '296' or dataseq == '1000':
                        datainfo = '1'
                        valueINF = 1
            elif match2:
                  datainfo = '1'
                  valueINF = 1
            elif match3:
                  datainfo = '1'
                  valueINF = 1
            elif match4:
                  datainfo = '1'
                  valueINF = 1
      elif datapro == '17':
            baca17=kolom[6]
            baca17len=kolom[5]
            match = re.search(r'1.3.6.1.2.1.1.1.0', baca17)
            match2 = re.search(r'205.206.231.10', baca17)
            match3 = re.search(r'342', baca17len)
            match4 = re.search(r'72', baca17len)


            match5 = re.search(r'[[].+[]]',baca17)
            if match5:
                  data17 = re.sub(r'\[','',match5.group())
                  data17 = re.sub(r',','',data17)
                  data17 = re.sub(r'\]','',data17)
            match6 = re.search(r'Malformed Packet', data17)

            if match:
                  datainfo = '1'
                  valueINF = 1
            if match2:
                  datainfo = '1'
                  valueINF = 1
            if match3:
                  datainfo = '1'
                  valueINF = 1
            if match4:
                  datainfo = '1'
                  valueINF = 1
            if match6:
                  datainfo = '1'
                  valueINF = 1

      datautama = dataip.split(';')
      datautama.append(datapro)
      datautama.append(datalength)
      datautama.append(dataflag)
```

```
        datautama.append(dataseq)
        datautama.append(datainfo)

        datatxt = '"'.split(';')
        datatxt.append(dataip)
        datatxt.append('", "')
        datatxt.append(datapro)
        datatxt.append('", "')
        datatxt.append(datalength)
        datatxt.append('", "')
        datatxt.append(dataflag)
        datatxt.append('", "')
        datatxt.append(dataseq)
        datatxt.append('", "')
        datatxt.append(datainfo)
        datatxt.append('", "')

        print '[IP SOURCE, PROTOCOL, LENGTH, FLAG, SEQ, INFO,
LABEL]'
        print datautama
        print ' '

        cur.execute('insert into uji
values(null,?,?,?,?,?,?,null)', datautama)
        con.commit()
```

Lampiran 2. Klasifikasi Naïve Bayes

```
import sqlite3 as lite
import math
import sys

con = lite.connect('length.db')

curL = con.cursor()
curL.execute('select count(*) from latih')
dataL = curL.fetchone()[0]
dataL = float(dataL)

curU = con.cursor()
curU.execute('select count(*) from uji')
dataU = curU.fetchone()[0]
dataU = int(dataU)

cur1 = con.cursor()
cur1.execute('select count(*) from latih where label=1')
data1 = cur1.fetchone()[0]
data1 = float(data1)

cur0 = con.cursor()
cur0.execute('select count(*) from latih where label=0')
data0 = cur0.fetchone()[0]
data0 = float(data0)

print ("===== ===== ===== ===== ===== ===== ")
print "Data latih  =",dataL
print "Data Uji    =",dataU
print "Serangan    =",data1
print "Normal      =",data0
print ("===== ===== ===== ===== ===== ===== ")

noUji = 1

for no in range(1,dataU+1):
    ### Menghitung Protocol
##############################################################
    curprouji = con.cursor()
    curprouji.execute("select PROTOCOL from uji where
NO=?", [noUji])
    dataprouji = curprouji.fetchone()[0]

    curpro1 = con.cursor()
    curpro1.execute("select count(*) from latih where
label=1 and PROTOCOL=?", [dataprouji])
    datapro1 = curpro1.fetchone()[0]
    datapro1 = float(datapro1)

    curpro0 = con.cursor()
    curpro0.execute("select count(*) from latih where
label=0 and PROTOCOL=?", [dataprouji])
```

```
      datapro0 = curpro0.fetchone()[0]
      datapro0 = float(datapro0)

      ### Menghitung Length
##############################################################
##
      curlenuji = con.cursor()
      curlenuji.execute("select LENGTH from uji where NO=?",
[noUji])
      datalenuji = curlenuji.fetchone()[0]
      datalenuji = float(datalenuji)

      curlen1 = con.cursor()
      curlen1.execute('select count(*) from latih where
label=1')
      datalen1 = curlen1.fetchone()[0]
      datalen1 = float(datalen1)

      curlen1.execute("select avg(LENGTH) from latih where
label=1")
      avglen1 = curlen1.fetchone()[0]
      avglen1 = float(avglen1)

      curlen1.execute('select LENGTH from latih where
label=1')
      datalen1b = curlen1.fetchall()

      riw3 = 0

      for row in datalen1b:
            n=0
            riw = row[n]-avglen1
            riw2 = riw * riw
            riw3 = riw3 + riw2
            n=n+1
      variansilen1=riw3/(datalen1-1)

      atas = datalenuji - avglen1
      atas2 = atas * atas
      bawah = 2 * variansilen1
      pangkate = atas2/bawah
      e = 2.71828
      kanan=math.pow(e,pangkate)

      kiri = 1/math.sqrt(6.28*variansilen1)
      densitaslen1 = kanan * kiri



      curlen0 = con.cursor()
      curlen0.execute('select count(*) from latih where
label=0')
      datalen0 = curlen0.fetchone()[0]
```

```
        datalen0 = float(datalen0)

        curlen0.execute("select avg(LENGTH) from latih where
label=0")
        avglen0 = curlen0.fetchone()[0]
        avglen0 = float(avglen0)

        curlen0.execute('select LENGTH from latih where
label=0')
        datalen0b = curlen0.fetchall()

        riw3 = 0

        for row in datalen0b:
            n=0
            riw = row[n]-avglen0
            riw2 = riw * riw
            riw3 = riw3 + riw2
            n=n+1
        variansilen0=riw3/(datalen0-1)

        atas = datalenuji - avglen0
        atas2 = atas * atas
        bawah = 2 * variansilen0
        pangkate = atas2/bawah
        e = 2.71828
        kanan=math.pow(e,pangkate)

        kiri = 1/math.sqrt(6.28*variansilen0)

        densitaslen0 = kanan * kiri

        #### Menghitung Flag
##############################################################
###
        curflauji = con.cursor()
        curflauji.execute("select FLAG from uji where NO=?",
[noUji])
        dataflauji = curflauji.fetchone()[0]

        curfla1 = con.cursor()
        curfla1.execute("select count(*) from latih where
label=1 and FLAG=?", [dataflauji])
        datafla1 = curfla1.fetchone()[0]
        datafla1 = float(datafla1)

        curfla0 = con.cursor()
        curfla0.execute("select count(*) from latih where
label=0 and FLAG=?", [dataflauji])
        datafla0 = curfla0.fetchone()[0]
        datafla0 = float(datafla0)
```

```
      ### Menghitung Sequence
##############################################################
      cursequji = con.cursor()
      cursequji.execute("select SEQUENCE from uji where
NO=?", [noUji])
      datasequji = cursequji.fetchone()[0]
      datasequji = float(datasequji)

      curseq1 = con.cursor()
      curseq1.execute('select count(*) from latih where
label=1')
      dataseq1 = curseq1.fetchone()[0]
      dataseq1 = float(dataseq1)

      curseq1.execute("select avg(SEQUENCE) from latih where
label=1")
      avgseq1 = curseq1.fetchone()[0]
      avgseq1 = float(avgseq1)

      curseq1.execute('select SEQUENCE from latih where
label=1')
      dataseq1b = curseq1.fetchall()

      riw3 = 0

      for row in dataseq1b:
            n=0
            riw = row[n]-avgseq1
            riw2 = riw * riw
            riw3 = riw3 + riw2
            n=n+1
      variansiseq1=riw3/(dataseq1-1)

      atas = datasequji - avgseq1
      atas2 = atas * atas
      bawah = 2 * variansiseq1
      pangkate = -(atas2/bawah)
      e = 2.71828
      kanan=math.pow(e,pangkate)

      kiri = 1/math.sqrt(6.28*variansiseq1)
      densitasseq1 = kanan * kiri



      curseq0 = con.cursor()
      curseq0.execute('select count(*) from latih where
label=0')
      dataseq0 = curseq0.fetchone()[0]
      dataseq0 = float(dataseq0)

      curseq0.execute("select avg(SEQUENCE) from latih where
label=0")
```

```
        avgseq0 = curseq0.fetchone()[0]
        avgseq0 = float(avgseq0)

        curseq0.execute('select SEQUENCE from latih where
label=0')
        dataseq0b = curseq0.fetchall()

        riw3 = 0

        for row in dataseq0b:
             n=0
             riw = row[n]-avgseq0
             riw2 = riw * riw
             riw3 = riw3 + riw2
             n=n+1
        variansiseq0=riw3/(dataseq0-1)

        atas = datasequji - avgseq0
        atas2 = atas * atas
        bawah = 2 * variansiseq0
        pangkate = -(atas2/bawah)
        e = 2.71828
        kanan=math.pow(e,pangkate)

        kiri = 1/math.sqrt(6.28*variansiseq0)

        densitasseq0 = kanan * kiri

        ### Menghitung Info
##############################################################
####
        curinfuji = con.cursor()
        curinfuji.execute("select INFO from uji where NO=?",
[noUji])
        datainfuji = curinfuji.fetchone()[0]

        curinf1 = con.cursor()
        curinf1.execute("select count(*) from latih where
label=1 and INFO=?", [datainfuji])
        datainf1 = curinf1.fetchone()[0]
        datainf1 = float(datainf1)

        curinf0 = con.cursor()
        curinf0.execute("select count(*) from latih where
label=0 and INFO=?", [datainfuji])
        datainf0 = curinf0.fetchone()[0]
        datainf0 = float(datainf0)

        ### Menghitung Class
##############################################################
###

        point1 = data1/dataL
```

```
        point0 = data0/dataL

        pointpro1 = datapro1/data1
        pointpro0 = datapro0/data0

        pointfla1 = datafla1/data1
        pointfla0 = datafla0/data0

        pointinf1 = datainf1/data1
        pointinf0 = datainf0/data0

        if pointtotal1 > pointtotal0:
              print noUji, "Probing"
              print "   Attack - Normal =", point1, "-", point0
              print "   Protocol        =", pointpro1, "-",
pointpro0
              print "   Length          =", densitaslen1, "-",
densitaslen0
              print "   Flag            =", pointfla1, "-",
pointfla0
              print "   Sequence        =", densitasseq1, "-",
densitasseq0
              print "   Info            =", pointinf1, "-",
pointinf0
              print "   Total           =", pointtotal1, "-",
pointtotal0
              update = "update uji set LABEL=1 where NO=?"
        else:
              print noUji, "Normal"
              print "   Attack - Normal =", point1, "-", point0
              print "   Protocol        =", pointpro1, "-",
pointpro0
              print "   Length          =", densitaslen1, "-",
densitaslen0
              print "   Flag            =", pointfla1, "-",
pointfla0
              print "   Sequence        =", densitasseq1, "-",
densitasseq0
              print "   Info            =", pointinf1, "-",
pointinf0
              print "   Total           =", pointtotal1, "-",
pointtotal0
              update = "update uji set LABEL=0 where NO=?"

        curUPD = con.cursor()
        curUPD.execute(update, [noUji])

        noUji = noUji + 1

curhtg1 = con.cursor()
curhtg1.execute("select count(*) from uji where label=1")
datahtg1 = curhtg1.fetchone()[0]
```

```
curhtg0 = con.cursor()
curhtg0.execute("select count(*) from uji where label=0")
datahtg0 = curhtg0.fetchone()[0]

print ""
print "Jumlah Serangan     =",datahtg1
print "Jumlah Normal  =",datahtg0
print ""

con.commit()
```

Lampiran 3. Menutup Koneksi

```
import sqlite3 as lite
import math
import sys
import re
import os
import subprocess

con = lite.connect('naive.db')
#con = lite.connect(sys.argv[2])

curIP = con.cursor()
curIP.execute('select IP from uji where label=1')
#dataIP = curIP.fetchone()
dataIP = curIP.fetchall()

curIPx = con.cursor()
curIPx.execute('select count(*) from uji where label=1')
dataIPint = curIPx.fetchone()[0]
dataIPint = int(dataIPint)

#print dataIPint

noIP = 0
noIPint = []

for no in dataIP:
     bacaip = no[0]

     if noIP == 0:
          print bacaip
          noIPint.append(bacaip)
          noIP=noIP + 1
          subprocess.call(["iptables","-A","INPUT","-
s",bacaip,"-j","DROP"])
     elif noIP > 0 and bacaip != noIPint[noIP-1]:
          print bacaip
          noIPint.append(bacaip)
          noIP=noIP + 1
          subprocess.call(["iptables","-A","INPUT","-
s",bacaip,"-j","DROP"])

con.commit()
```