VISUALISASI LALU LINTAS DATA PADA JARINGAN DENGAN MENGGUNAKAN LIBRARY SHARPPCAP

Agustinus Noertjahyana [1)], Haryanto Tunggary [2)], Justinus Andjarwirawan [3)]
Teknik Informatika, Universitas Kristen Petra
agust@peter.petra.ac.id, m26406155@john.petra.ac.id, justin@peter.petra.ac.id

Abstract

Network technology is developing very rapidly nowadays. Almost all computers connected to the network. Therefore, a network administrator is required to constantly monitor network activity in order to detect any irregularities in the network as early as possible. By using the library for packet capture (library used by packet sniffer to capture packets on the network), can be made an application that can capture packets will then be visualized to show data that is easy to understand.

The aim of this project is to create applications that can monitor data packets both incoming and outgoing network, interpreting, displaying such data in forms that are easier to understand (visualize it), and also at the same time monitor the packages so it can detect the discrepancy (the virus). Application is using VB.Net programming language, Library SharpPcap, SQL Server, ClamAV, and Snort.

The test results showed that the application system can detect the virus and intrusion, it can capture incoming and outgoing packets on an interface in real time, and can visualize it using line charts, pie charts, column charts, and also the timeline.

Keywords— Packet Capture, Packet Sniffer, Virus Detection

1. PENDAHULUAN

Tak dapat dipungkiri bahwa teknologi jaringan komputer berkembang sangat pesat dimasa sekarang ini. Hampir semua komputer terhubung ke jaringan. Bahkan hampir semua aplikasi sudah meminta koneksi ke jaringan baik untuk melakukan autentikasi (*register*), *update*, dan lain sebagainya.

Terkadang seorang *administrator* jaringan dituntut untuk dapat memonitor paket-paket yang terdapat didalam jaringannya baik paket yang masuk maupun paket yang keluar dari jaringan. Hal ini dilakukan agar seorang *administrator* jaringan dapat tetap memonitor keadaan jaringan yang berada di bawah tanggung jawab administrasinya.

Cara untuk memonitor keadaan jaringan sangat mungkin dilakukan dengan adanya aplikasi *packet sniffer*. *Packet sniffer* digunakan untuk melihat paket-paket yang keluar maupun masuk kedalam jaringan. Namun masih dibutuhkan *skill* dari seorang *administrator* jaringan agar penggunaan *packet sniffer* ini bisa bekerja secara optimal sehingga dapat melihat paket yang diinginkan.

Dengan memanfaatkan *library* untuk *packet capture* (*library* yang digunakan oleh *packet sniffer* untuk menangkap paket di jaringan), maka pada penelitian ini dikembangkan sebuah aplikasi yang dapat menangkap paket data yang keluar dan masuk serta memvisualisasikannya, sehingga dapat menampilkan data yang menarik, tidak membingungkan, dan mudah untuk dimengerti.

Diharapkan dengan adanya aplikasi ini dapat membantu pekerjaan seorang *administrator* jaringan sebab tidak perlu lagi melihat data-data yang membingungkan dan susah dimengerti.

2. METODE PENELITIAN

Metode penelitian yang dilakukan pada penelitian ini meliputi : studi literature tentang *packet sniffer*, *packet capture* serta *library* yang digunakan dalam aplikasi, melakukan analisis dan desain aplikasi serta melakukan pengujian aplikasi.

2.1 Packet Sniffer

Agar suatu aplikasi dapat melihat paket-paket data baik yang masuk maupun keluar jaringan dibutuhkan sebuah *packet sniffer* [1]. Dengan *packet sniffer*, dapat dilihat paket pada jaringan dalam detail *real-time*. Dengan *Ethernet*, dapat dilihat *traffic* komputer. *Ethernet* adalah *bus* jaringan. Ketika satu mesin mengirim sebuah paket ke mesin lain pada sebuah *Ethernet*, paket

dapat dilihat di sepanjang kabel *Ethernet*.

Dengan, menaruh *card Ethernet* dalam mode khusus yang disebut *Promiscuous Mode*, *card* akan melewatkan paket ke sistem operasi, tidak peduli alamat tujuan dari paket. Sistem operasi kemudian akan melewatkan paket ke aplikasi *packet sniffer* [1].

2.1.1 Promiscuous Mode

Modus *promiscuous* atau *mode promisc* adalah konfigurasi kartu jaringan yang membuat kartu meloloskan semua *traffic* jaringan yang diterima ke *kernel*, bukan hanya ditujukan ke *frame (Fitur* yang biasanya digunakan untuk mengendus paket), dan jaringan dijembatani untuk virtualisasi *hardware*.

Setiap *frame* termasuk perangkat keras *MAC address* (*Media Access Control*) bila kartu jaringan menerima *frame*, biasanya membuangnya kecuali jika *frame* ditujukan ke kartu itu. Dalam modus *promiscuous*, kartu itu memungkinkan semua *frame* diloloskan, sehingga memungkinkan komputer untuk membaca *frame* yang ditujukan untuk mesin lain atau perangkat jaringan.

Banyak sistem operasi membutuhkan hak akses *superuser* untuk mengaktifkan modus *promiscuous*. Sebuah simpul *non-routing* dalam modus *promiscuous* umumnya hanya dapat memonitor *traffic* ke dan dari *node* lain di dalam *collision domain* yang sama (untuk *Ethernet* dan *Wireless LAN*) atau *ring* (untuk *Token Ring* atau *FDDI*). Komputer pada jaringan *hub* yang sama memenuhi kebutuhan ini, oleh karena itu jaringan *switch* digunakan untuk melawan penggunaan berbahaya mode *promiscuous*. *router* dapat memantau semua lalu lintas yang dirutekan olehnya.

Promiscuous mode sering digunakan untuk mendiagnosa masalah jaringan konektivitas. Ada beberapa program yang memanfaatkan fitur ini untuk menunjukkan pengguna semua data yang di-*transfer* melalui jaringan. Beberapa protokol seperti *FTP* dan *Telnet transfer* data dan *password* dalam bentuk *clear* teks, tanpa enkripsi, dan *scanner* jaringan dapat melihat data ini. Oleh karena itu, pengguna komputer disarankan untuk menjauh dari protokol tidak aman seperti menggunakan *telnet* dan yang lebih aman seperti *SSH*.

2.1.2 SharpPcap

SharpPcap adalah *packet capture framework* untuk *.NET environment* yang didasarkan dari komponen *WinPcap* yang terkenal [2]. Tujuan dari *library* ini adalah untuk menyediakan sebuah *API* untuk menangkap, memasukkan, menganalisis dan membangun paket menggunakan bahasa *.NET* Seperti *C #* dan *VB.NET*.

Berikut adalah fitur yang saat ini didukung oleh *SharpPcap* [1] :

- Menghitung dan menampilkan rincian tentang *interface* jaringan fisik pada mesin *Windows*.
- Menangkap paket jaringan tingkat rendah yang melalui *interface* yang diberikan.
- Menganalisis dan melakukan *parsing* protokol berikut: *Ethernet, ARP, IP, TCP, UDP, ICMP, IGMP*.
- Memasukkan paket jaringan tingkat rendah pada *interface* tertentu.
- Penanganan (membaca dan menulis) *file* paket *offline*.
- Memasukkan paket menggunakan Antrian Kirim.
- Mengumpulkan statistik jaringan pada antarmuka jaringan yang diberikan.

Pada *SharpPcap* v3.0.0 *API* menandakan perubahan besar dari seri 2.x. Semua *class parsing* paket *SharpPcap* telah dihapus. Fungsi paket *parsing* telah diganti dengan *Packet.Net* v0.3.0. Alasan untuk memisahkan fungsi paket *parsing* menjadi *library* yang terpisah adalah untuk menghapus fungsi yang tidak terkait dengan *pcap* dari *SharpPcap* untuk tujuan modularitas dan untuk meninjau kembali rancangan paket *parsing* kode.

2.1.3 Intrusion Detection System

Intrusion Detection digunakan untuk mengidentifikasi akses individu yang menggunakan sebuah sistem komputer tanpa hak. Percobaan untuk masuk secara paksa juga harus teridentifikasi. *Intrusion Detection System* (*IDS*) mengatasi masalah intrusi dengan mengamati *traffic* jaringan pada *medium* atau pada *host* tertentu, mencari pola dari kejadian atau request yang mungkin mengindikasikan sebuah penyerangan. *IDS* memiliki *database* dari *attack signature* (pola aktivitas yang sudah dilihat oleh manajer jaringan sebelumnya).

Pola ini disimpan di dalam *database* dalam sebuah bahasa spesifikasi yang sederhana. Sebagai contoh, sebuah *signature* untuk *worm Sircam* (yang dibawa oleh *e-mail*) pada sebuah *ISD* adalah:

alert TCP $EXTERNAL any - > $INTERNAL 80 (msg: “IDS552/web-iis_IISISAPI Overflow ida”; dsize: > 239, flags: A+, content: “.ida?”;).

Dilihat dari cara kerja dalam menganalisa apakah paket data dianggap sebagai penyusupan atau bukan, *IDS* dibagi menjadi 2 [3]:

1. *Knowledge Based* atau *Misuse Detection*

Knowledge-Based IDS dapat mengenali adanya penyusupan dengan cara menyadap paket data kemudian membandingkannya dengan *database rule IDS* (berisi catatan paket serangan). Jika paket data mempunyai pola yang sama dengan salah satu atau lebih pola di *database rule IDS*, maka paket tersebut dianggap sebagai serangan, dan demikian juga sebaliknya, jika paket data tersebut sama sekali tidak mempunyai pola yang sama dengan pola di *database rule IDS*, maka paket data tersebut dianggap bukan serangan.

2. *Behavior Based* atau *Anomaly Based*.

Behavior Based (Anomaly) dapat mendeteksi adanya penyusupan dengan mengamati adanya kejanggalan-kejanggalan pada sistem, atau adanya penyimpangan-penyimpangan dari kondisi normal, sebagai contoh ada penggunaan *memory* yang melonjak secara terus menerus atau ada koneksi parallel dari 1 buah *IP* dalam jumlah banyak dan dalam waktu yang bersamaan. Kondisi-kondisi tersebut dianggap kejanggalan yang kemudian oleh *IDS* jenis *anomaly based* dianggap sebagai serangan.

Sedangkan dilihat dari kemampuan mendeteksi penyusupan pada jaringan, *IDS* dibagi menjadi 2 yakni: *host based* dan *network based*. *Host based* mampu mendeteksi hanya pada *host* tempat implementasi *IDS*, sedangkan *network based IDS* mampu mendeteksi seluruh *host* yang berada satu jaringan dengan *host* implementasi *IDS* tersebut.

2.2 Virus komputer

Perkembangan virus sudah terjadi sejak tahun 1983 [4]. Menurut catatan, pada tahun tersebut telah ada virus yang bernama *Elk Cloner* yang menyerang *Apple* dengan metode penyebaran melalui *Flash Disk*. Tidak diketahui siapa pembuat virus ini. Pada tahun yang sama, Fred Cohen memberikan sebuah seminar yang berjudul *"Computer Virus-Theory and Experiments"* yang menunjukkan teori serta demonstrasi tentang bagaimana virus komputer bisa diciptakan.

Semenjak itu, perkembangan virus terus terjadi seperti epidemi. Muncul sebuah virus yang dibuat di Pakistan yang berhasil menyerang MS DOS kemudian virus *Lehigh* yang juga menyerang MS DOS dengan menginfeksi file '*Command.com*' yang merupakan file wajib untuk sistem operasi MS DOS. Pada tahun 1988, virus *Machintos* yang pertama lahir dengan nama *MacMag* dan pada tahun yang sama pula, jaringan internet sempat mendapatkan gangguan hebat akibat dari *worm* yang diciptakan oleh Robert Moris.

Untuk menghindari deteksi, virus mulai berevolusi seperti layaknya sebuah virus yang mampu mengubah dirinya sendiri. Virus *polymorphic* pertama diperkenalkan di Switzerland dengan virus yang dinamakan *Tequila*. Pada tahun yang sama, yaitu 1992, *Dark Avenger Mutation Engine* (DAME) diperkenalkan dan merupakan sebuah engine pertama yang mampu mengubah segala jenis virus menjadi *polymorphic* dan pada tahun ini pula diperkenalkan *Virus Creation Laboratory* (VCL) yang menjadi sebuah software pembuat virus yang pertama di dunia.

Jika sebelumnya virus selalu menyerang *file-file executable* seperti *.COM*, *.EXE*, dan *.SYS*, kali ini virus mulai melakukan penyerangan terhadap *file* data yang sebelumnya tidak pernah menjadi sasaran. Virus *Laroux* dan *Staog* merupakan virus *macro* pertama yang menyerang *Microsoft Excel*. Virus *macro* dilanjutkan oleh kemunculan virus *Melissa* yang memanfaatkan *Microsoft Word* sebagai media penyebaran yang berhasil menginfeksi jutaan komputer-komputer didunia serta dilanjutkan lagi oleh virus *macro Corner* yang menginfeksi *Microsoft Project*.

Penyebaran dan penyerangan virus semakin menyebar dan pada tahun 2000, bahkan ditemukan sebuah virus bernama *Timofonica* yang menyerang sistem telepon yang dibuat dengan *Visual Basic Script*. Pada tahun ini juga, penyerangan virus terjadi pada PDA *Palm* yang sangat populer pada masa itu dan virus ini dinamakan sebagai virus *Liberty*.

Kemampuan virus terus berevolusi. Jika sebelumnya sebuah virus hanya bisa menyerang *Windows* atau *Linux*, kini muncul virus bernama *Wimux* yang mampu menyerang sistem operasi *Windows* dan *Linux* sekaligus. Pada tahun 2002, epidemi terjadi pada *worms* dengan kemunculan *Sharp-A*, *SQLSpyder*, *Sobig*, *Slammer*, *Lovgate*, *Fizzer*, *Blaster*, dan lain sebagainya.

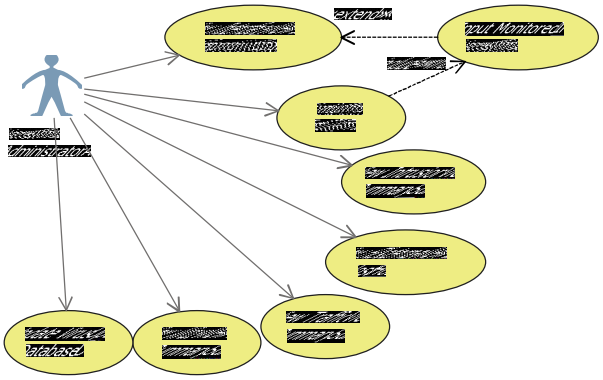
Virus telah menjadi epidemi yang sampai saat ini pun terus bermutasi dengan kemampuan-kemampuan dan daya rusak yang besar. Berbagai virus membuat berita-berita baru setiap harinya seperti layaknya seorang bintang besar yang tidak pernah pudar popularitasnya [5].

3. DESAIN

Tahapan desain pada aplikasi ini terdiri dari 7 fungsi utama yaitu :

- View Interface Information
- Monitor Traffic
- View Intrusion Summary
- View Traffic Summary
- View Virus Summary
- Update Virus Database
- View Eksternal Host

Untuk lebih jelas tentang fungsi utama dapat dilihat pada *Use Case Diagram* pada gambar 1.



Gambar 1. Use Case Diagram

3.1 View Interface Information

Fungsi dimana *user* dapat melihat informasi detail mengenai *Network Interface Card (NIC)* pada komputer. Pada fungsi ini juga disediakan fungsi sehingga *user* dapat meng-*input* atau mengganti jaringan yang ingin di-monitor oleh *interface* tersebut dengan meng-*input*-kan *IP Address* dan *CIDR*-nya.

3.2 Monitor Traffic

Monitor *Traffic* adalah fungsi utama dari aplikasi ini. Fungsi ini mulai dilaksanakan saat *user* telah memilih *interface* mana yang ingin di-monitor. Pada fungsi ini, aplikasi akan me-monitor dan menganalisa setiap paket yang masuk ke dalam jaringan. Dalam fungsi ini juga terdapat fungsi *IDS* dan virus *detection*.

3.3 View Intrusion Summary

View Intrusion Summary adalah fungsi dimana *user* dapat melihat ringkasan mengenai intrusi-intrusi yang terjadi. Pada fungsi ini, awalnya aplikasi akan menampilkan statistik mengenai intrusi berdasarkan harian, mingguan, bulanan, dan tahunan. Setelah itu *user* dapat melihat data intrusi yang terjadi secara lebih detail berdasarkan rentang waktu yang telah dipilih oleh *user*.

Setelah itu akan ditampilkan *pie chart* yang menggambarkan perbandingan antara paket yang normal dan paket yang dianggap intrusi, *timeline* yang menggambarkan saat-saat intrusi terjadi, *log* yang mencatat semua intrusi yang terjadi, data dan penjelasan mengenai intrusi yang terjadi, dan *host-host* yang menyerang dan yang diserang berdasarkan *IP Address*.

3.4 View Traffic Summary

View Traffic Summary adalah fungsi dimana *user* dapat melihat ringkasan mengenai *traffic* jaringan secara umum. Pada fungsi ini, awalnya akan ditampilkan statistik mengenai *traffic* jaringan selama ini berdasarkan harian, mingguan, bulanan, dan tahunan.

Setelah itu, *user* dapat melihat data *traffic* jaringan secara lebih detail berdasarkan rentang waktu yang telah dipilih oleh *user*. Kemudian akan ditampilkan *pie chart* yang yang menggambarkan perbandingan paket berdasarkan *protocol*, dan statistik paket selama rentang waktu tersebut berdasarkan *protocol*.

3.5 View Virus Summary

View Virus Summary adalah fungsi dimana *user* dapat melihat ringkasan mengenai virus-virus yang menyerang jaringan. Pada fungsi ini, awalnya akan ditampilkan statistik virus yang masuk ke dalam jaringan berdasarkan harian, mingguan, bulanan, dan tahunan.

Setelah itu, *user* dapat melihat *history* secara lebih detail berdasarkan rentang waktu yang dipilih oleh *user* yang memperlihatkan data mengenai virus-virus apa saja yang memasuki jaringan dalam rentang waktu tersebut.

3.6 Update Virus Database

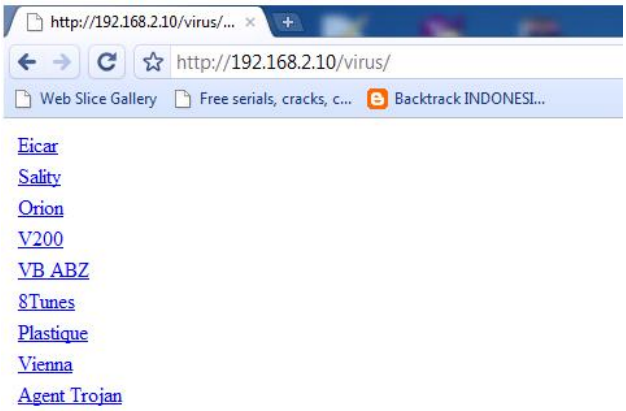
Update Virus Database adalah fitur dimana *user* dapat meng-*update database* virus yang digunakan untuk mendeteksi adanya virus yang memasuki jaringan.

4. HASIL DAN PEMBAHASAN

Pembahasan terhadap hasil penelitian dan pengujian terhadap aplikasi ini dilakukan dengan menggunakan 1 unit komputer serta sebuah *virtual* komputer yang akan digunakan sebagai komputer yang akan menjadi penyerang pada jaringan dan juga sebagai *server* yang menjadi tempat *Hosting file-file virus* yang akan di-*download*. Kedua komputer tersebut dihubungkan menggunakan *Microsoft Loopback Adapter*.

Pengujian terhadap aplikasi dan interface dilakukan dengan cara menjalankan aplikasi dan kemudian men-*trigger* agar aplikasi memberikan *respond*. Cara untuk men-*trigger* adalah dengan membuka *web* yang telah di-*hosting* pada *virtual* komputer, men-*download file* pada *web* tersebut untuk menguji *virus detection*, dan melakukan

penetration testing untuk menguji *Intrusion detection system*. Tampilan *web* yang di-*hosting* pada *virtual* komputer dapat dilihat pada Gambar 2.

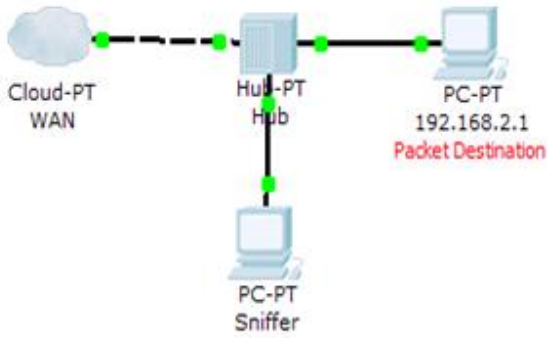


Gambar 2. Contoh *Website*

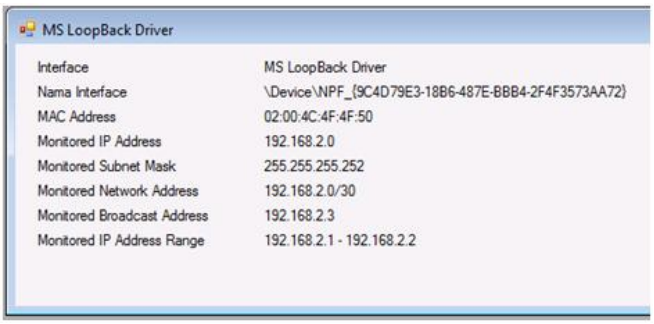
4.1 Pengujian Packet Sniffing

Pengujian dilakukan dengan cara menentukan *Network* yang akan diawasi oleh aplikasi. *IP Address* pada komputer di-*set* 192.168.2.1 sedangkan *IP Address* 192.168.2.10. Dengan settingan *IP Address* tersebut, maka *packet-packet* yang ditangkap hanya yang menuju maupun dari *IP Address* 192.168.2.1.

Pada percobaan pertama, *Network* yang akan diawasi di-*set* 192.168.2.0/30 sehingga Paket yang diawasi adalah Paket yang berasal maupun menuju *IP Address* 192.168.2.1-192.168.2.2. Ilustrasi dari jaringan yang dapat dilihat pada Gambar 3, sedangkan informasi dari *interface* yang dimonitor dapat dilihat pada Gambar 4.

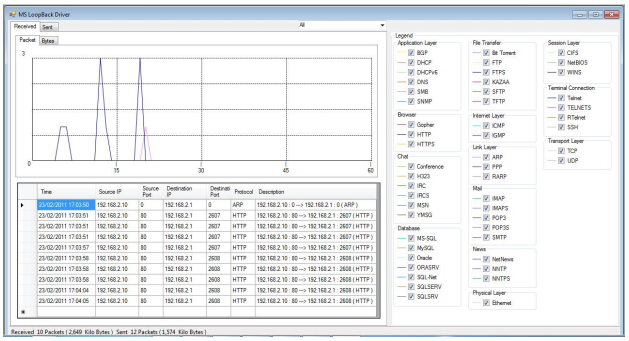


Gambar 3. Ilustrasi Jaringan



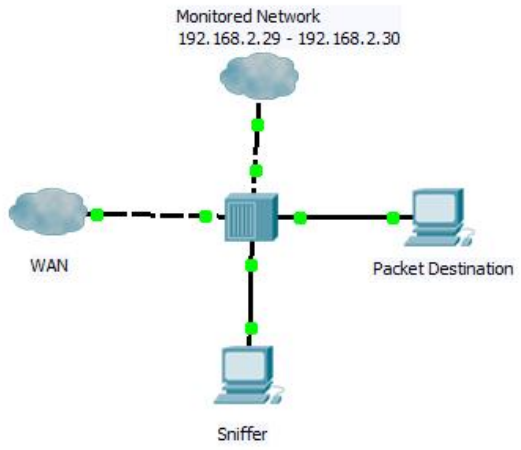
Gambar 4. *Interface Information*

Setelah itu aplikasi dijalankan untuk mencoba menangkap paket. Paket-paket yang masuk akan di-*trigger* dengan membuka *website* pada *virtual* komputer, sehingga paket-paket HTTP akan mulai dikirimkan ke komputer. Hasilnya akan ditampilkan pada Gambar 5.

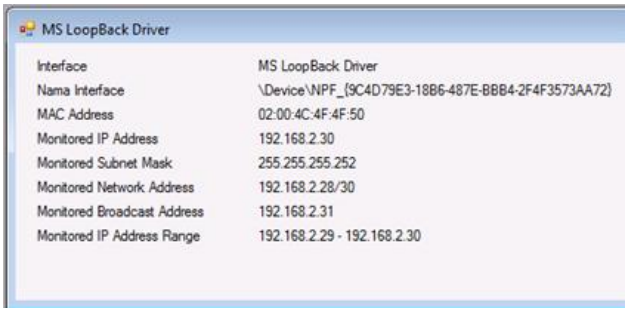


Gambar 5. *Sniffing Packet*

Ternyata paket-paket data ditampilkan baik paket yang masuk maupun paket yang dikirim oleh *interface* tersebut. Pada percobaan kedua, *Monitored Network* akan diganti menjadi 192.168.2.30/30. Desain *Interface* yang baru akan ditampilkan pada Gambar 6, dan *interface information* yang baru akan ditampilkan pada Gambar 7.

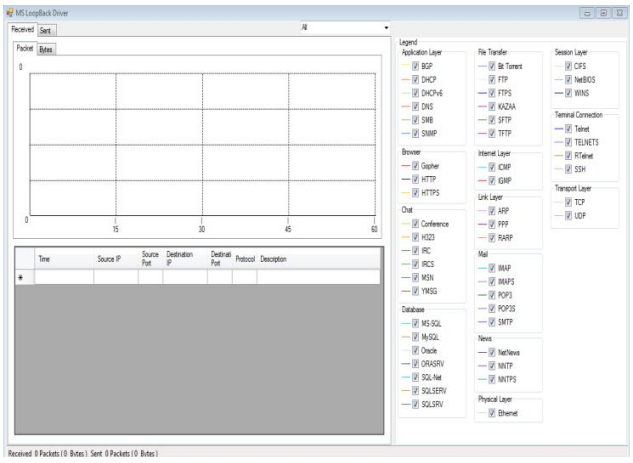


Gambar 6. Ilustrasi Jaringan Baru



Gambar 7. *Interface Information* Baru

Setelah itu aplikasi dijalankan untuk mencoba menangkap paket. Paket-paket yang masuk akan di-*trigger* dengan membuka *website* pada *virtual* komputer, sehingga paket-paket *HTTP* akan mulai dikirimkan ke komputer. Hasilnya akan ditampilkan pada Gambar 8.



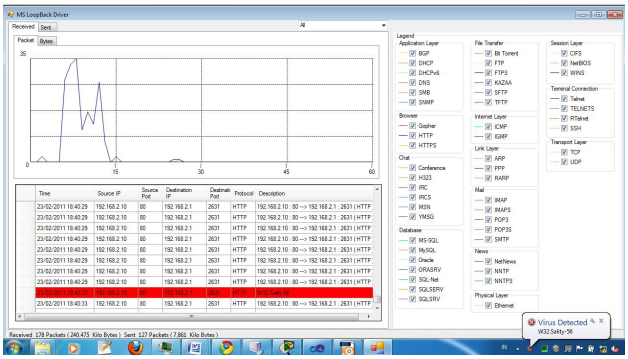
Gambar 8. *Sniffing Packet*

Ternyata setelah lebih dari lima menit terus men-*trigger* dengan membuka *website*, tidak ada satu paketpun yang ditangkap oleh aplikasi yang ditampilkan. Hal ini disebabkan karena paket yang masuk tidak ditujukan bagi *IP Address* dari *range* 192.168.2.29-192.168.2.30. oleh sebab itu, walaupun paket telah ditangkap namun paket tersebut di-*drop* oleh aplikasi sebab tidak sesuai dengan *network* yang dimonitor.

4.2 Pengujian Deteksi Virus

Pengujian dilakukan dengan mencoba men-*download* beberapa sampel virus yang telah disediakan. Sampel virus tersebut telah disediakan pada *website* yang telah di-*hosting* pada *virtual* komputer. Virus yang disediakan adalah *Eicar, Sality, Orion, V200, Trojan.Dropper, 8 Tunes, Plastique, Vienna*, dan *Agent Trojan*.

Setelah *file* selesai di-*download*, *file* akan di-*cek* apakan *corrupt* atau tidak. Cara untuk mengetahui *corrupt* atau tidaknya adalah dengan mencoba untuk membuka *file* tersebut (dalam hal ini di-*extract* sebab *file-file* tersebut dalam bentuk *rar*). Jika *file* berhasil dibuka, maka *file* dinyatakan tidak *corrupt* sedangkan kalau tidak berarti *file* dinyatakan *corrupt*. Saat aplikasi mendeteksi virus akan ditampilkan seperti Gambar 9.



Gambar 9. Aplikasi Mendeteksi Virus

Selain ditampilkan pada *datagrid* dengan warna merah, aplikasi juga akan menampilkan *notification* yang menuliskan nama virus yang terdeteksi. Tampilan *notification* akan ditampilkan seperti Gambar 10.

Gambar 10. *Notification* Saat Virus Terdeteksi

Hasil uji coba dengan menggunakan metode *download* biasa dan dengan metode *partial download*, akan ditampilkan dalam Tabel 1 dan Tabel 2.

Tabel 1. Hasil Uji Coba Menggunakan Metode *Download* Biasa

Nama File	File Size	File Corrupt	Virus Terdeteksi
Eicar.rar	186 Bytes	Tidak	Ya
Sality.rar	224 KB	Tidak	Ya
Orion.rar	424 Bytes	Tidak	Ya
V200.rar	269 Bytes	Tidak	Ya
VB ABZ.rar	2,42 MB	Ya	Ya
8 Tunes.rar	1,7 KB	Tidak	Ya
Plastique.rar	7,4 KB	Tidak	Ya
Vienna.rar	2,34 KB	Tidak	Ya
Agent Trojan.rar	113 KB	Tidak	Ya

Tabel 2. Hasil Uji Coba Menggunakan Metode *Partial Download*

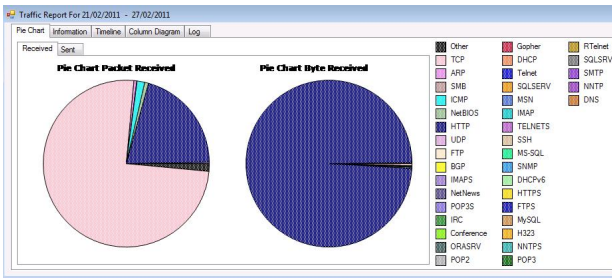
Nama File	File Size	File Corrupt	Virus Terdeteksi
Eicar.rar	186 Bytes	Tidak	Ya
Sality.rar	224 KB	Ya	Ya
Orion.rar	424 Bytes	Tidak	Ya
V200.rar	269 Bytes	Tidak	Ya
VB ABZ.rar	2,42 MB	Ya	Tidak
8 Tunes.rar	1,7 KB	Tidak	Ya
Plastique.rar	7,4 KB	Tidak	Ya
Vienna.rar	2,34 KB	Tidak	Ya
Agent Trojan.rar	113 KB	Ya	Ya

Virus di-*download* dengan menggunakan fitur *download* biasa pada *browser*, dan menggunakan *download manager*. Pada *download* biasa, ternyata terdapat 1 *file* yang *corrupt* sedangkan jika men-*download* dengan *download manager* (*partial download*) ternyata terdapat 3 *file* yang *corrupt*.

4.3 Pengujian Traffic Report

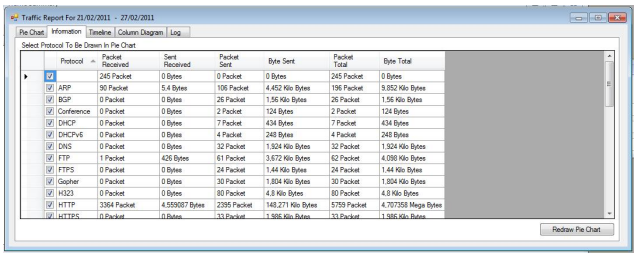
Traffic Report dapat dilihat dengan cara meng-*click datagrid* untuk memilih tanggal atau *range* tanggal yang telah ditampilkan dalam *datagrid*. Selain itu, kita juga dapat melihat *traffic Report* dengan menentukan sendiri *range* tanggal pada *tab Custom*.

Traffic Report memiliki lima *tab* utama, yaitu *Pie Chart*, *Information*, *Timeline*, *Column* Diagram, dan *Log*. Pada *Tab Pie Chart* akan ditampilkan *Pie Chart* yang menggambarkan persentase dari aktivitas jaringan yang dibedakan berdasarkan *Protocol* selama *range* waktu yang ditentukan. Tampilan *Pie Chart* dapat dilihat pada Gambar 11.



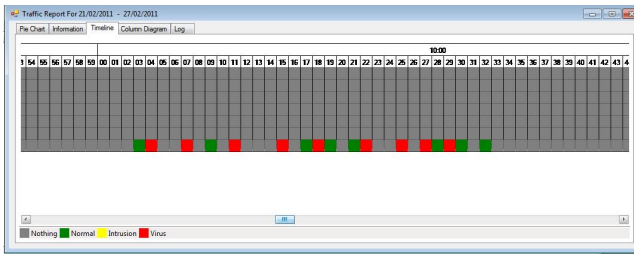
Gambar 11. Traffic Report (Pie Chart)

Tab kedua adalah *Information Tab*. *Information Tab* akan menampilkan data aktivitas jaringan berdasarkan *Protocol*. Data yang ditampilkan adalah *Protocol*, jumlah paket yang masuk, keluar, dan total keduanya, juga jumlah *byte* yang masuk, keluar, dan total keduanya. Selain menampilkan data, kita juga dapat memilih *Protocol* apa saja yang mau digambarkan dalam *Pie Chart*. *Protocol* yang dicentang akan digambarkan jika tombol di-*click*. Gambar *Tab Information* akan ditampilkan pada Gambar 12.

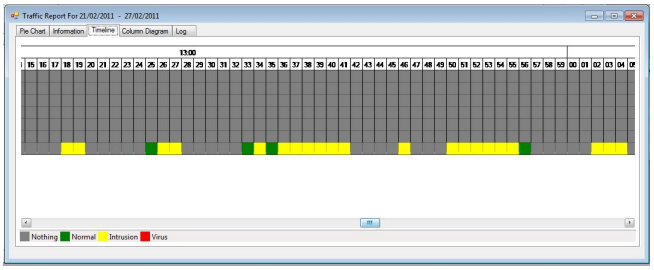


Gambar 12. *Traffic Report* (*Information Tab*)

Tab ketiga adalah *Timeline Tab*. Pada *Tab* ini, akan ditampilkan *timeline* yang akan menampilkan saat-saat kapan saja intrusi, dan virus terdeteksi. Pada *timeline* ini, terdapat 4 macam warna, yaitu abu-abu yang menggambarkan saat tidak ada aktivitas jaringan, hijau yang menggambarkan adanya aktivitas jaringan biasa, kuning yang menggambarkan ada intrusi yang terdeteksi, dan yang terakhir warna merah yang menggambarkan ada virus yang terdeteksi. Gambar saat virus terdeteksi akan ditampilkan pada Gambar 13, dan gambar saat intrusi terdeteksi akan ditampilkan pada Gambar 14.

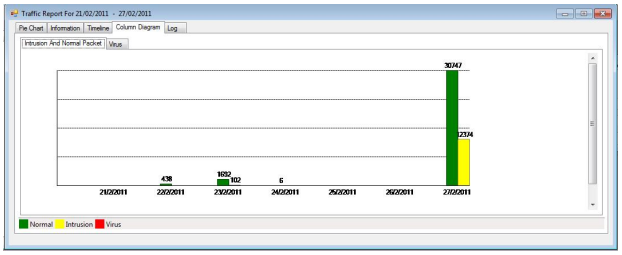


Gambar 13. *Traffic Report* (*Timeline Tab*-virus)

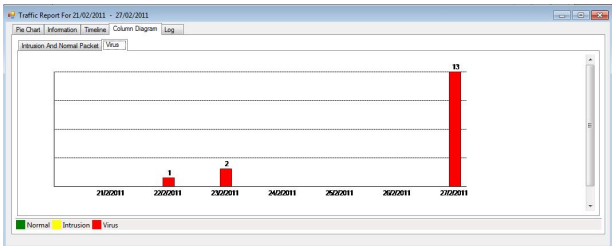


Gambar 14. *Traffic Report* (*Timeline Tab*-intrusi)

Tab keempat adalah *Column* Diagram. Pada *Tab* ini, akan ditampilkan diagram batang dengan menggunakan satuan tertentu. Diagram batang yang digambarkan ada dua yaitu virus, dan intrusi dan normal *packet*. Gambar diagram batang intrusi dan normal paket akan ditampilkan pada Gambar 15, sedangkan diagram batang virus akan ditampilkan pada Gambar 16.

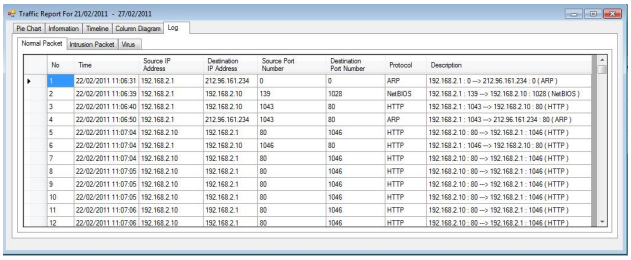


Gambar 15. Diagram Batang Paket Intrusi dan Paket Normal



Gambar 16. Diagram Batang Virus

Tab terakhir adalah *Log*. Pada *Tab Log* akan ditampilkan semua catatan mengenai aktivitas *traffic* dan *event* yang terjadi. *Traffic* dan *event* tersebut kemudian akan dibedakan lagi menjadi data *traffic* normal, *traffic* intrusi, dan juga virus yang terdeteksi. *Tab Log* akan ditampilkan pada Gambar 17.



Gambar 17. *Traffic Report* (*Log Tab*)

5. KESIMPULAN

Kesimpulan yang dapat diambil pada penelitian ini adalah :

- Aplikasi hanya mendeteksi virus pada *file* yang masuk ke dalam jaringan melalui *protocol* HTTP.
- Hasil pendeteksian virus, sangat bergantung dari hasil *file* yang disadap. Semakin mirip *file* hasil sadapan dengan *file* asli (tidak *corrupt*), maka makin baik hasil pendeteksiannya.
- Virus hanya dapat dideteksi jika aplikasi melakukan *sniffing packet* dari awal proses *file* tersebut di-*download* hingga selesai.
- Untuk mendapatkan *file* yang sama persis, diperlukan pemahaman mengenai *dissector* paket. Selain *dissector*, diperlukan juga pemahaman mengenai jaringan, dan *protocol* yang digunakan untuk men-*transfer file*.
- *File* yang berisi virus walaupun terdeteksi, tidak dapat dihalangi untuk memasuki jaringan (*File* tetap masuk kedalam jaringan).
- Aplikasi mampu untuk mendeteksi adanya virus pada *file* yang *corrupt*.

6. SARAN

Saran-saran yang dapat digunakanuntuk penelitian lebih lanjut adalah :

- Dapat dikembangkan lebih lanjut agar aplikasi dapat mendukung IPv6.
- Menambah jumlah *protocol* yang didukung oleh aplikasi.
- Memperbaiki kemampuan untuk meng-*cloning file* yang di-*download* dengan menggunakan teknik *partial download* dan juga *download* biasa menggunakan *sliding windows* agar tidak ada lagi *file cloning* yang *corrupt*.
- Meningkatkan kemampuan aplikasi dalam mendeteksi virus agar dapat mendeteksi virus dengan lebih cepat lagi.

DAFTAR PUSTAKA

[1] Gal, Tamir. (2010). *"SharpPcap" tamir gal | sharpPcap*. Retrieved April 5, 2010, from http://www.tamirgal.com/blog/page/SharpPcap.aspx.

[2] Mansfield, Niall. (2004). *Practical TCP/IP mendesain, menggunakan, dan troubleshooting jaringan TCP/IP di linux dan windows* Jilid 1. (Prabantini Dwi, Trans.). Yogyakarta: ANDI.

[3] Mansfield, Niall. (2004). *Practical TCP/IP mendesain, menggunakan, dan troubleshooting jaringan TCP/IP di linux dan windows* Jilid 2. (Prabantini Dwi, Trans.). Yogyakarta: ANDI.

[4] Raf Knowledge. (2010). *Trik memonitor jaringan*. Jakarta: PT Elex Media Komputindo.

[5] Slameto, A.A . (2010). *Sistem pencegah penyusupan*. Retrieved August 10, 2010, from http://www.scribd.com/doc/36389197/01-STMIK-AMIKOM-Yogyakarta-Sistem-Pencegah-Penyusupan.