PENGANTAR

Alhamdulillah, segala puji bagi Allah SWT atas rizki-Nya sehingga penyusun dapat menyelesaikan tugas akhir yang berjudul "**Perancangan Sistem Pemfilteran Layanan Internet Pada *Gateway* Dengan Menggunakan Squidguard**" dengan baik.

Tugas akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang.

Tak banyak yang bisa penyusun sampaikan kecuali ungkapan terima kasih, syukur, dan doa kepada berbagai pihak yang telah dengan tulus ikhlas memberikan bimbingan, arahan, dan dukungan hingga penulisan tugas akhir ini dapat terselesaikan. Pada kesempatan kali ini, dengan segala kesungguhan dan rasa rendah hati, penyusun mengucapkan banyak terima kasih kepada:

1. Bapak dan Ibu yang selalu mendoakan dan mendukung dalam menyelesaikan tugas akhir ini.
2. Keluarga Besar yang turut mendoakan dan mendukung dalam menyelesaikan tugas akhir ini.
3. Bapak Ir. Primantara Hari Trisnawan selaku Dosen Pembimbing yang telah banyak memberikan masukan dan koreksi dalam penyusunan tugas akhir ini;
4. Bapak Raden Arief Setiawan, ST, MT selaku Dosen Pembimbing yang telah banyak memberikan bimbingan dan arahan terhadap penyusunan tugas akhir ini.
5. Ketua dan Sekretaris Jurusan Teknik Elektro dan segenap Staf Pengajar, Administrasi, dan Perpustakaan Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya;
6. Welly Purnomo atas segala bantuan, dukungan dan saran yang banyak untuk penyusun.
7. Raden Ardianto Bimo Nugroho atas segala bantuan, dukungan dan saran yang banyak untuk penyusun.
8. Rieke Adriati Wijayanti, ST, M. Desdrianton Islamy, Rudi Rahadian, Wawan Rachmanto, Heri Muryanto, ST yang telah banyak memberikan bantuan, saran, dan dukungan kepada penyusun.

9. Ir. Kasyful Amron dan Reza Andria Siregar, ST yang telah memberikan kepercayaan, kesempatan dan dukungan kepada penyusun untuk menyelesaikan tugas akhir ini.
10. Rekan-rekan di UPPTI – UB (Unit Pengkajian dan Penerapan Teknologi Universitas Brawijaya) khususnya rekan-rekan di Divisi CNS (*Computer Network Service*) yang telah memberikan kesempatan dan dukungan kepada penyusun untuk menyelesaikan tugas akhir ini.
11. Rekan-rekan di TPTI - FT (Tim Pengembangan Teknologi Informasi Fakultas Teknik) yang telah memberikan kesempatan dan dukungan kepada penyusun untuk menyelesaikan tugas akhir ini.
12. Rekan-rekan Asisten di Laboratorium Sistem Informasi atas dukungan dan bantuan dalam penelitian dan pengembangan program;
13. Rekan-rekan mahasiswa Teknik Elektro Fakultas Teknik Universitas Brawijaya khususnya rekan-rekan angkatan 2001 [*Duracell*] yang turut membantu kelancaran penyusunan tugas akhir ini
14. Semua pihak yang tidak dapat penyusun sebutkan satu-persatu di sini, sehingga dapat terselesaikannya tugas akhir ini.

Semoga Allah SWT memberikan balasan kebahagiaan dan kesuksesan atas segala budi baik yang telah diberikan kepada penyusun.

Penyusun menyadari bahwa tugas akhir ini masih banyak kekurangannya dan masih jauh dari sempurna, untuk itu saran dan kritik yang membangun sangat penyusun harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 14 Mei 2007

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ABSTRAK

HANS FIANTONY DWIYANTO. 2007 : *Perancangan Sistem Pemfilteran Layanan Internet pada Gateway dengan Menggunakan SquidGuard.* Skripsi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Mei 2007. Dosen Pembimbing: Ir. Primantara Hari Trisnawan dan Raden Arief Setiawan, ST, MT.

Seiring dengan pesatnya teknologi Internet, saat ini telah terdapat banyak situs di Internet dengan berbagai macam isi. Situs-situs berkategori dewasa pun, ada banyak sekali sehingga perlu adanya filter agar situs-situs seperti ini tidak diakses pada tempat-tempat tertentu, misalnya kantor. Untuk dapat memenuhi kebutuhan tersebut, salah satu alternatifnya adalah dibuat suatu sistem untuk memfilter layanan Internet pada *gateway* dengan menggunakan SquidGuard.

SquidGuard adalah suatu *plug-in* dari Squid yang merupakan program kombinasi antara *filter*, *redirector,* dan *access controller* yang terpasang pada suatu *gateway* dengan sistem operasi Fedora Core 6. Sistem pemfilteran ini diimplementasikan pada suatu jaringan komputer bertopologi *Star* dengan tujuan untuk memfilter layanan Internet.

Pengujian sistem pemfilteran layanan internet pada *gateway* dengan menggunakan squidguard digunakan untuk mengetahui apakah sistem ini telah dapat melakukan pemblokiran terhadap situs-situs berkategori dewasa dan pemblokiran untuk me-*download file-file* multimedia pada saat jam kerja.

Pengujian pertama dilakukan pada *internal network* untuk mengetahui apakah semua *client* dan *gateway* telah dapat saling berkomunikasi.

Pengujian kedua dilakukan pada *Gateway* untuk mengetahui apakah paket data yang berasal dari *internal network* dapat diteruskan oleh komputer *gateway* menuju *external network.*

Pengujian ketiga dilakukan pada Squid untuk mengetahui apakah Squid dapat meneruskan paket data dari komputer *client* dan meneruskan paket data balasan kembali ke komputer *client* tanpa melakukan *cache web content.*

Pengujian keempat dilakukan pada SquidGuard secara umum pada saat jam kerja, maksudnya adalah SquidGuard diuji langsung dengan komputer *client* menggunakan *browser.*

Pengujian kelima dilakukan pada SquidGuard secara umum di luar jam kerja, maksudnya adalah SquidGuard diuji langsung dengan komputer *client* menggunakan *browser*.

Hasil pengujian secara keseluruhan adalah bahwa SquidGuard berhasil memfilter paket data terlarang pada saat terlarang dan meneruskan paket data apapun pada saat yang bukan terlarang.

Kata Kunci : Filter, Blokir, Situs, *Download*, Multimedia, Squid, SquidGuard, *Gateway*, Internet, Jaringan Komputer, Fedora Core 6.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan suatu teknologi yang bernama Internet[1], saat ini telah maju sedemikian pesatnya. Perkembangan ini juga tidak lepas dari semakin banyaknya situs yang dapat diakses oleh pengguna sejak ditemukannya internet[2]. Situs-situs di Internet saat ini telah sedemikian banyaknya mulai dari pendidikan, forum, profil, sampai hal-hal yang kurang layak untuk diakses di tempat umum. Oleh karena itu, dalam menggunakan internet perlu adanya pembatasan atas hal-hal tertentu tergantung pihak atau lembaga yang memberikan layanan internet.

Masalah yang dihadapi oleh pihak penyedia layanan internet, misalnya pihak perusahaan atau pimpinan suatu lembaga adalah bagaimana menerapkan suatu pembatasan layanan akses internet sehingga semua pengguna internet patuh pada aturan yang telah ditetapkan. Untuk mengatasi hal tersebut maka perlu adanya suatu penerapan sistem pembatasan akses yang dapat menangani kebutuhan-kebutuhan seperti pembatasan akses ke situs-situs tertentu, pembatasan akses pada pengguna-pengguna tertentu, atau pembatasan akses pada waktu-waktu tertentu.

Beberapa *software* yang dapat digunakan untuk menjalankan fungsi-fungsi yang telah disebutkan di atas antara lain adalah SquidGuard, DansGuardian, dan URLfilterDB. DansGuardian merupakan sebuah *software* yang mampu melakukan pemfilteran dengan berbagai metode yang meliputi mulai dari *domain* hingga ekstensi *file* [BAR-07]. URLfilterDB adalah sebuah basis data URL yang digunakan oleh suatu *web proxy server* untuk memfilter suatu *web content* yang tidak diinginkan [URL-07]. SquidGuard adalah suatu *software* tambahan (*plug-in*) pada Squid yang digunakan untuk pemfilteran berdasarkan sebuah daftar filter dengan cepat [BAL-02].

Diantara ketiga *software* yang disebutkan diatas, SquidGuard merupakan salah satu pilihan terbaik dengan beberapa keunggulan dibandingkan dengan *software* lainnya.Keunggulan-keunggulan tersebut antara lain adalah SquidGuard merupakan suatu program kombinasi antara *filter* (penyaring dengan ekspresi tertentu), *redirector*

[1] Internet (dengan “I” besar) adalah sistem komputer umum, yang berhubung secara global dan menggunakan TCP/IP sebagai protokol pertukaran paket data (packet switching communication protocol)

[2] internet (dengan “i” kecil) adalah rangkaian komputer yang saling berhubungan membentuk suatu jaringan komputer yang besar

(meneruskan suatu *url*[3] ke *url* yang lain), dan *access controller* (suatu pengontrol hak akses). SquidGuard memiliki karakteristik sangat fleksibel, cepat, mudah proses instalasi-nya, *portable,* dan gratis karena berlisensi GNU

SquidGuard memerlukan dukungan dari sistem yang menjadi pondasinya, dalam arti sistem tempat SquidGuard itu dijalankan. Untuk itu, perlu satu unit komputer dengan spesifikasi cukup tinggi untuk berfungsi sebagai *gateway*[4] dan server Squid & SquidGuard, PC tersebut perlu diberi suatu sistem operasi. Sistem Operasi yang digunakan adalah Fedora Core 6 yang mana adalah salah satu distribusi linux berbasis RedHat yang berlisensi GNU General Public License. Dengan dukungan penuh dari RedHat dan komunitas yang cukup besar, Fedora Core 6 merupakan suatu sistem operasi yang handal dan stabil untuk menjalankan fungsi sebagai *gateway* maupun *server*.

Karena SquidGuard adalah program *plug-in* bagi Squid maka instalasi Squid adalah suatu keharusan. Squid merupakan suatu program *web-cache proxy* handal dengan berbagai macam fungsi yang dapat dikombinasikan mulai dari mempercepat pengaksesan *web server* dengan melakukan *cache* pada *request*, melakukan *cache* pada web agar dapat diakses lebih cepat oleh para pengguna yang berada dalam jaringan komputer yang sama, sampai membantu mengamankan keamanan jaringan komputer lokal. Squid bekerja pada sistem operasi berbasis Unix dan telah dikembangkan selama bertahun-tahun hingga diakui sebagai sistem yang stabil.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dikhususkan pada :

1. Merancang Sistem Pemfilteran layanan Internet yang terdiri dari Sistem *Gateway*, Sistem Squid, dan Sistem SquidGuard.
2. Merancang dan mengkonfigurasi Sistem *Gateway.*
3. Merancang dan mengkonfigurasi Sistem Squid yang tidak melakukan *web cache* , untuk menjalankan Sistem SquidGuard pada Sistem *Gateway.*

---

[3] url (*Uniform Resource Locator*) adalah rangkaian karakter menurut suatu format standar tertentu, yang digunakan untuk menunjukkan alamat suatu sumber seperti dokumen dan gambar di Internet.

[4] *Gateway* adalah sebuah perangkat yang digunakan untuk menghubungkan satu jaringan komputer dengan satu atau lebih jaringan komputer yang menggunakan protokol komunikasi yang berbeda sehingga informasi dari satu jaringan computer dapat diberikan kepada jaringan komputer lain yang protokolnya berbeda

4. Merancang dan mengkonfigurasi Sistem SquidGuard untuk membatasi layanan akses layanan Internet secara tepat pada Sistem *Gateway*.
5. Menguji Sistem *Gateway*, Sistem Squid, dan Sistem SquidGuard yang telah dirancang dan dikonfigurasi.

1.3 Ruang Lingkup Pembahasan

Dalam perencanaan dan pembuatan skripsi ini perlu dilakukan pembatasan masalah. Pembatasan masalah yang diajukan dalam skripsi ini antara lain:

1. Sistem operasi yang digunakan adalah Fedora Core 6 dan program-program yang digunakan adalah program dengan standar Fedora Core 6.
2. Perangkat lunak yang digunakan untuk memfilter layanan internet pada gateway adalah Squid dan SquidGuard sebagai *plug-in*.
3. Paket-paket data yang difilter adalah paket-paket data dari sisi dalam (*intranet*) yang menuju ke sisi luar (*internet*).
4. Paket-paket data yang difilter adalah paket-paket data yang menggunakan protokol HTTP.
5. Masalah yang dibahas adalah perancangan suatu sistem untuk memfilter layanan internet pada *gateway* dengan menggunakan SquidGuard

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah merancang suatu sistem untuk memfilter layanan internet pada *gateway* dengan menggunakan SquidGuard.

1.5 Sistematika Penulisan

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup permasalahan, tujuan penulisan serta sistematika penulisan.

BAB II Tinjauan Pustaka

Membahas kajian pustaka dan teori dasar mengenai Protokol TCP/IP, Topologi Jaringan Komputer, IPTables, Squid, SquidGuard, dan Protokol HTTP.

BAB III Metodelogi Penulisan

Membahas metodelogi yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan dan pembuatan sistem, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan Sistem

Membahas perancangan sistem yang sesuai dengan teori yang ada.

BAB V Implementasi Sistem

Membahas implementasi sistem yang telah dirancang.

BAB VI Pengujian Sistem

Membahas pengujian dan analisis sistem yang telah diimplementasikan.

BAB VII Kesimpulan dan Saran

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

BAB II
TINJAUAN PUSTAKA

Bab ini akan menjelaskan tinjauan pustaka dan dasar teori yang menunjang penulisan skripsi mengenai perancangan sistem pemfilteran layanan internet pada *gateway* dengan menggunakan SquidGuard.

Kajian pustaka yang dijelaskan tentang Sistem Pemfilteran meliputi hal-hal sebagai berikut : perlunya pembuatan Sistem Pemfilteran dan komponen-komponen pembentuk Sistem Pemfilteran. Berdasarkan kajian pustaka tersebut maka dasar teori yang digunakan adalah protokol TCP/IP, Topologi Jaringan, Iptables, Squid, SquidGuard, dan Protokol HTTP.

2.1. Perlunya Pembuatan Sistem Pemfilteran

Sistem Pemfilteran dalam kaitannya dengan layanan akses Internet adalah kumpulan dari komponen-komponen perangkat keras dan perangkat lunak yang membentuk suatu sistem untuk melakukan proses pemisahan antara paket data – paket data yang mengalir antara dua titik yang berkomunikasi dalam satu jaringan internet [IES-03]. Sistem Pemfilteran dibutuhkan untuk melakukan pemblokiran terhadap *web content* yang tidak diinginkan terkait dengan kebijakan yang berlaku di suatu tempat. Pemblokiran yang dapat dilakukan meliputi pemblokiran akses situs, pemblokiran aktivitas *download*, pembatasan akses pengguna layanan akses Internet. Dalam hubungannya dengan penerapan kebijakan di suatu tempat, Sistem Pemfilteran dapat digunakan untuk memblokir situs-situs berkategori dewasa, membatasi aktivitas *download file* multimedia, dan membatasi hak akses karyawan.

2.2. Komponen-Komponen Pembentuk Sistem Pemfilteran

Komponen-komponen pembentuk Sistem Pemfilteran meliputi perangkat keras dan perangkat lunak. Perangkat keras dalam sistem ini, meliputi topologi jaringan komputer secara fisik dan perangkat fisik *gateway*.

Topologi jaringan secara fisik terdiri dari topologi *bus*, *ring*, dan *star*. Penjelasan yang lebih terperinci dari topologi-topologi tersebut dapat dilihat di bagian dasar teori. *Gateway* secara fisik dapat dibedakan dalam dua hal, yaitu *dedicated* dan komputer yang difungsikan sebagai *gateway*. *Gateway* yang *dedicated* adalah suatu perangkat yang memang dirancang dan dibuat untuk melakukan proses *routing*

sebagai fungsi utamanya. *Gateway dedicated* banyak dibuat oleh vendor-vendor perangkat jaringan terkemuka seperti Cisco dan Nortel. Karakteristik dari *gateway dedicated* ini adalah stabil, mampu memproses aliran data dalam jumlah yang besar, memiliki kinerja yang sangat bagus, mahal, dan perlu keahlian khusus mengkonfigurasi untuk setiap model dan merk [SUG-07].

Komputer yang difungsikan sebagai *gateway* merupakan sebuah komputer yang dikonfigurasi secara khusus untuk dapat menjalankan fungsi sebagai *gateway.* Karakteristik dari *gateway* model ini adalah murah, mudah dikonfigurasi, fleksibel, kurang stabil, kinerja yang cukup bagus untuk model jaringan komputer yang tidak terlalu besar, dan kemampuannya tergantung dari spesifikasi masing-masing komponen komputernya.

Perangkat lunak dalam Sistem Pemfilteran terdiri dari Sistem Squid dan Sistem SquidGuard. Squid merupakan suatu program *web-cache proxy* handal dengan berbagai macam fungsi yang dapat dikombinasikan mulai dari mempercepat pengaksesan *web server* dengan melakukan *cache* pada *request,* melakukan *cache* pada web agar dapat diakses lebih cepat oleh para pengguna yang berada dalam jaringan komputer yang sama, sampai membantu mengamankan keamanan jaringan komputer lokal. Squid digunakan oleh hampir semua *server* berbasis UNIX di dunia. Dengan pertimbangan akan keunggulan-keunggulan yang dimilki Squid tersebut, maka dalam dalam perancangan yang akan dibuat nanti, akan difokuskan pada Squid.

Untuk melakukan membantu Squid dalam proses pemfilteran, terdapat beberapa *software plug-in* yang digunakan, antara lain SquidGuard, DansGuardian, dan URLfilterDB. DansGuardian merupakan sebuah *software* yang mampu melakukan pemfilteran dengan berbagai metode yang meliputi mulai dari *domain* hingga ekstensi *file* [BAR-07]. DanGuardian dapat digunakan pada Squid yang berjalan di lingkungan sistem Linux, FreeBSD, OpenBSD, NetBSD, Mac OS X, HP-UX, dan Solaris. Meskipun *software* ini gratis, beberapa layanan di DansGuardian ditujukan untuk versi yang tidak gratis.

URLfilterDB adalah sebuah basis data URL yang digunakan oleh suatu *web proxy server* untuk memfilter suatu *web content* yang tidak diinginkan [URL-07]. URLfilterDB memiliki tiga metode untuk melakukan pemblokiran, yaitu *content scanning, artificial intelligence,* dan *blacklist.* Masing-masing metode tersebut memiliki keunggulan tersendiri tergantung dari biaya, kemampuan pengguna,

banyaknya pemblokiran, dan banyaknya jumlah pengguna. Meskipun mengklaim *software* ini murah, *software* ini tidak gratis.

SquidGuard adalah suatu *software* tambahan (*plug-in*) pada Squid yang digunakan untuk pemfilteran berdasarkan sebuah daftar filter dengan cepat [BAL-02]. SquidGuard memiliki karakteristik sangat fleksibel, cepat, mudah proses instalasi-nya, *portable*, dan gratis. Dengan banyaknya kemampuan dan keunggulan yang dimilki oleh SquidGuard, terlebih lagi dengan lisensinya yang GNU *Public License* maka dalam perancangan yang akan dibuat nanti, akan difokuskan pada SquidGuard.

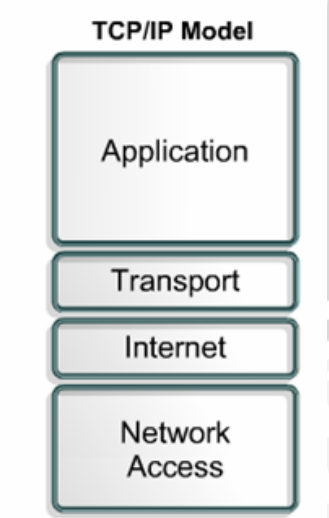
2.3. *Transmission Control Protocol/Internet Protocol* (TCP/IP)

Protokol TCP/IP memiliki beberapa komponen yaitu Arsitektur Protokol TCP/IP, Enkapsulasi Data, dan *Routing*.

2.3.1. Arsitektur Protokol TCP/IP

TCP/IP adalah kumpulan dari protokol yang setiap protokol bertanggungjawab atas bagian-bagian tertentu dari komunikasi data. TCP/IP terdiri dari empat lapis kumpulan protokol yang bertingkat [WIL–93]. Keempat lapisan atau *layer* tersebut dari yang teratas, sebagaimana yang diperlihatkan dalam Gambar 2.1 adalah :

- *Application Layer*
- *Transport Layer*
- *Internet Layer*
- *Network Interface Layer*



Gambar 2.1 Model TCP/IP
Sumber : [CIS-04]

2.3.1.1. *Application Layer*

Pada lapisan teratas inilah letak semua aplikasi yang menggunakan protokol TCP/IP, antara lain :

a. FTP (*File Transfer Protocol*)

FTP merupakan salah satu aplikasi TCP/IP yang digunakan untuk memindahkan data dari komputer yang satu ke komputer yang lain secara efisien.

b. SMTP (*Simple Mail Transport Protocol*) atau E-Mail

SMTP memberikan cara yang mudah dan cepat dalam mengirim informasi. SMTP ini pada awalnya hanya merupakan salah satu jenis transfer file.

c. HTTP (*Hypertext Transfer Protocol*)

Protokol ini digunakan untuk jenis layanan *World Wide Web* atau layanan akses situs pada jaringan TCP/IP.

d. DNS (*Domain Name Service*)

Protokol ini digunakan untuk memetakan nama-nama host ke alamat-alamat jaringannya.

2.3.1.2. *Transport Layer*

Transport layer merupakan lapisan komunikasi data yang mengatur aliran data antara dua host, untuk keperluan aplikasi lapisan di atasnya. Ada dua buah protokol pada lapisan ini, yaitu TCP dan UDP.

a. **TCP (*Transmission Control Protocol*)**

TCP merupakan protokol yang menyediakan layanan yang berkarakteristik sebagai berikut : *connection oriented*, *reliable*, dan *byte stream service*. *Connection oriented* berarti sebelum melakukan pertukaran data, dua aplikasi pengguna TCP harus melakukan pembentukan hubungan (*handshake*) terlebih dahulu. *Reliable* (dapat diandalkan) berarti TCP menerapkan proses deteksi kesalahan paket data dan *re-transmission* (transmisi ulang). *Byte stream service* berarti paket data dikirimkan ke tujuan secara berurutan [PUR-01:55].

+	Bits 0 - 3	4 - 9	10 - 15	16 - 31
0	Source Port			Destination Port
32	Sequence Number			
64	Acknowledgement Number			
96	Data Offset	Reserved	Flags The IP Header	Window
128	Checksum			Urgent Pointer
160	Options (optional)			
160/ 192+	Data			

Gambar 2.2 Format Segmen TCP

Sumber : [ISI-81a:15]

Seperti yang ditunjukkan dalam Gambar 2.2, segmen TCP terdiri atas beberapa field. *Source* dan *Destination Port* adalah *field* berisi angka yang mengidentifikasi aplikasi pengirim dan penerima segmen TCP. *Sequence Number* berisi nomor urut *byte stream* dalam data aplikasi yang dikirim. Setiap kali data ini sukses dikirim, pihak penerima data mengisi *field Acknowledgement Number* dengan S*equence Number* berikutnya yang diharapkan penerima.

Data offset berisi 4-bit *field* yang menunjukkan ukuran *header* TCP dalam 32-bit *words*. Ukuran *header* minimum adalah 5 *words* dan maksimum 15 *words*, yang memberikan besar minimum 20 *byte* dan maksimum 60 *byte*.

Reserved besarnya adalah 6-bit dan sebaiknya diisi dengan nol. *Field* ini digunakan untuk keperluan di masa depan.

Flags the IP header berisi 6 bit *flag*, yaitu : URG yang menunjukkan *Urgent Pointer field* penting, ACK yang menunjukkan *Acknowledgement field* penting, PSH yang menunjukkan *Push Function*, RST untuk melakukan setting ulang koneksi, SYN yang menunjukkan *Synchronize sequence numbers*, dan FIN yang menunjukkan tidak ada data lagi dari pengirim.

Field window diisi dengan panjang *window* (semacam *buffer*) penerimaan segmen TCP, merupakan banyak byte maksimal yang bisa diterima setiap saat. Lebar field ini adalah 16 bit (2 byte), sehingga nilai maksimalnya adalah 65535.

Checksum berfungsi untuk memeriksa apakah terdapat kesalahan pada *header* dan data.

Urgent Pointer berukuran 16-bit yang berfungsi mengimbangi *sequence numbers* yang menunjukkan *byte* terakhir dari *urgent data*, jika *flag* URG dipasang.

Options merupakan *field* tambahan yang dapat diikuti dengan *urgent data*. Jika *field* ini ada, maka total lebar *field* ini harus kelipatan dari 32-bit *words* dan pengaturan *field Data Offset* akan menyesuaikan.

Data bukan merupakan bagian dari *header* dan berisi apapun yang berasal dari *layer* diatasnya.

b. **UDP (*User Datagram Protocol*)**

UDP merupakan protokol *transport* yang sederhana. Berbeda dengan TCP yang *connection oriented*, UDP bersifat *connectionless*. Dalam UDP tidak ada *sequencing* (pengurutan kembali) paket data yang datang, *acknowledgement* terhadap paket data yang datang, atau transmisi ulang jika paket mengalami masalah di tengah jalan. Kemiripan UDP dengan TCP ada pada penggunaan *port number*. Sebagaimana digunakan pada TCP, UDP menggunakan *port number* ini untuk membedakan pengiriman *datagram* ke beberapa aplikasi berbeda yang terletak pada komputer yang sama. Karena sifatnya yang *connectionless* dan *unreliable*, UDP umumnya dipakai untuk mengirim data yang memerlukan kecepatan tetapi kurang peka terhadap kesalahan, seperti mengirim suara dan video. UDP ini bersifat *broadcasting* atau *multicasting*. Pengiriman *datagram* ke banyak *client* sekaligus akan efisien jika prosesnya menggunakan metode *connectionless*.

+	Bits 0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Gambar 2.3 Format *Datagram* UDP

Sumber : [POS-80:1]

Dalam Gambar 2.3, ditunjukkan format dari datagram UDP. Source Port dan Destination Port memiliki fungsi yang sama seperti pada TCP.

Datagram Length berisi panjang datagram, sedangkan Checksum berisi angka hasil perhitungan matematis yang digunakan untuk memeriksa kesalahan data.

2.3.1.3. *Internet Layer*

Internet Layer berfungsi untuk memungkinkan *host* mengirimkan paket data ke jaringan dan memungkinkan paket-paket data itu berjalan sendiri-sendiri ke tujuannya (yang besar kemungkinan berada di jaringan lain). Paket-paket data ini mungkin tiba di tujuan dengan urutan yang berbeda dengan urutan saat dikirimkan. Dalam kasus seperti ini, layer-layer yang berada di atasnya bertugas untuk mengatur kembali, bila pengiriman terurut diinginkan.

a. **IP (*Internet Protocol*)**

IP yang akan dibahas pada sub bab ini adalah Internet Protocol versi 4 (IPv4). Protokol IP merupakan inti dari protokol TCP/IP. Seluruh data yang berasal dari protokol pada layer di atas IP diolah oleh protokol IP, dan dipancarkan sebagai paket IP agar sampai ke tujuan. IP memiliki sifat yang dikenal sebagai *unreliable, connectionless,* dan *datagram delivery service.*

Unreliable berarti bahwa protokol IP tidak menjamin bahwa paket yang dikirim pasti sampai ke tujuan. Protokol IP hanya berjanji ia akan melakukan usaha sebaik-baiknya (*best effort delivery service*), agar paket yang dikirim tersebut sampai ke tujuan. Jika di perjalanan paket tersebut mengalami hal-hal yang tidak diinginkan (misalnya : salah satu jalur putus, *router* mengalami kongesti, atau *host/network* tujuan sedang *down*), maka protokol IP hanya memberitahukan pengirim paket melalui protokol ICMP (*Internet Control Messege Protocol*), bahwa terjadi masalah dalam pengiriman paket IP ke tujuan. Jika menginginkan kehandalan yang lebih baik, kehandalan itu harus disediakan oleh protokol yang berada di atas layer IP ini (yaitu TCP dan aplikasi pengguna).

Connectionless berarti dalam mengirim paket dari tempat asal ke tujuan, pihak pengirim dan penerima paket IP sama sekali tidak mengadakan perjanjian (*handshake*) terlebih dahulu.

Datagram delivery service berarti setiap paket data yang dikirim tanpa tergantung terhadap paket data yang lain. Akibatnya, jalur yang ditempuh oleh masing-masing paket data IP ke tujuannya bisa jadi berbeda satu dengan lainnya. Karena jalur yang ditempuh berbeda, kedatangan paket pun bisa jadi tidak berurutan.

Metode-metode diatas dipakai dalam pengiriman paket IP untuk menjamin agar paket IP tetap sampai di tujuan, walaupun salah satu jalur ke tujuan itu mengalami masalah.

+	Bits 0 - 3	4 - 7	8 - 15	16 - 18	19 - 31
0	Version	Header length	Type of Service (now DiffServ and ECN)	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
160/ 192+	Data				

Gambar 2.4 Format *Header* IPv4

Sumber : [ISI-81b]

Seperti yang terlihat dalam Gambar 2.4, Setiap paket IPv4 membawa data yang terdiri dari :

- *Version* (4 bit), berisi versi dari protokol IP yang dipakai, yaitu versi 4.
- *Header Length* (4 bit), berisi panjang dari header paket IP ini dalam hitungan 32 *bit word*.
- *Type of Service* (8 bit), berisi kualitas *service* yang dapat mempengaruhi cara penanganan paket IP ini.
- *Total Length of Datagram* (16 bit), berisi panjang IP *datagram* total dalam ukuran byte.
- *Identification* (16 bit), berisi nomor urutan yang bersama-sama dengan alamat sumber, alamat tujuan, dan protokol *user*, digunakan untuk mengidentifikasi sebuah *datagram* secara khusus, sehingga nomor ini dikhususkan untuk alamat sumber *datagram*, alamat tujuan, dan protokol *user* selama datagram tetap pada internet.
- *Flag* (3 bit), hanya dua bit yang sudah ditetapkan. Bit *More* digunakan untuk proses fragmentasi dan *reassembly*. Bit *Don't Fragment* mencegah terjadinya proses fragmentasi. Bit ini digunakan apabila telah diketahui bahwa tujuannya tidak memiliki kemampuan untuk memasang kembali fragment-fragment. Bagaimanapun juga, bila bit ini diset, datagram akan

dibuang apabila melebihi ukuran maksimum jaringan akhir. Oleh karena itu, bila bit ini diset, disarankan untuk menggunakan *routing* sumber untuk menghindari jaringan dengan ukuran paket maksimum.

- *Fragment Offset* (13 bit), menunjukkan di bagian mana datagram asli *fragment* ini termasuk, diukur dalam unit-unit 64 bit. Secara tidak langsung hal ini menyatakan fragment-fragment harus berisi *field* data yang panjangnya merupakan kelipatan 64 bit.
- *Time to Live* (8 bit), berisi jumlah *router* / *hop* maksimal yang boleh dilewati paket IP. Setiap kali paket IP melewati satu *router*, isi dari *field* ini dikurangi satu. Jika TTL telah habis dan paket tetap belum sampai ke tujuan, paket ini akan dibuang dan *router* terakhir akan mengirimkan paket ICMP *time exceeded*. Hal ini dilakukan untuk mencegah paket IP terus menerus berada di dalam *network*.
- *Protocol* (8 bit), mengandung angka yang mengidentifikasikan protokol *layer* atas pengguna isi data dari paket IP ini.
- *Header Checksum* (16 bit), berisi nilai *checksum* yang dihitung dari seluruh *field* dari *header* paket IP. Sebelum dikirimkan, protokol IP terlebih dahulu menghitung *checksum* dari *header* paket IP tersebut untuk nantinya dihitung kembali di sisi penerima. Jika terjadi perbedaan, maka paket ini dianggap rusak dan dibuang.
- IP address pengirim dan penerima data (masing-masing 32 bit), berisi alamat pengirim paket dan penerima paket.
- Beberapa byte *option*, diantaranya :
  - *Strict Source Route*. Berisi daftar lengkap IP *address* dari *router* yang harus dilalui oleh paket ini dalam perjalanannya ke *host* tujuan. Selain itu paket balasan atas paket ini, yang mengalir dari *host* tujuan ke *host* pengirim, diharuskan melalui *router* yang sama.
  - *Loose Source Route*. Dengan mengeset *option* ini, paket yang dikirim diharuskan singgah di beberapa *router* seperti yang disebutkan dalam *field option* ini. Jika diantara kedua *router* yang disebutkan terdapat *router* lain, paket masih diperbolehkan melalui *router* tersebut.

b. **ICMP (*Internet Control Message Protocol*)**

ICMP adalah protokol yang bertugas mengirimkan pesan-pesan kesalahan dan kondisi lain yang memerlukan perhatian khusus. Pesan atau paket ICMP dikirim jika terjadi masalah pada *layer* IP dan *layer* atasnya (TCP/UDP).

Ada dua tipe pesan yang dapat dihasilkan oleh ICMP yaitu *ICMP Error Message* dan *ICMP Query Message*. *ICMP Error Message*, sesuai namanya, dihasilkan jika terjadi kesalahan pada jaringan. Sedangkan *ICMP Query Message* adalah jenis pesan yang dihasilkan oleh protokol ICMP jika pengirim paket menginginkan informasi tertentu yang berkaitan dengan kondisi jaringan.

ICMP Error Message dibagi menjadi beberapa jenis, diantaranya :

- *Destination Unreachable*. Pesan ini dihasilkan oleh *router* jika pengiriman paket mengalami kegagalan akibat putusnya jalur, baik secara fisik maupun secara *logic*. *Destination Unreachable* dibagi menjadi beberapa tipe yang penting, yaitu :
  - *Network Unreachable,* jika jaringan tujuan tak dapat dihubungi.
  - *Host Unreachable,* jika *host* tujuan tak bisa dihubungi.
  - *Protocol at Destination is Unreachable,* jika di tujuan tak tersedia protokol tersebut.
  - *Port is Unreachable,* jika tidak ada *port* yang dimaksud pada tujuan.
  - *Destination Host is Unknown*, jika *host* tujuan tidak diketahui.
  - *Destination Network is Unknown*, jika *network* tujuan tidak diketahui.
- *Time Exceeded*. Paket ICMP jenis ini dikirimkan jika isi *field* TTL dalam paket IP sudah habis dan paket belum juga sampai ke tujuannya. Setiap kali sebuah paket IP melewati satu *router*, nilai TTL dalam paket tersebut dikurangi satu. TTL ini diterapkan untuk mencegah timbulnya paket IP yang terus menerus berputar-putar di *network* karena suatu kesalahan tertentu, sehingga menghabiskan sumber daya jaringan yang ada.

  Field TTL ini pula yang digunakan oleh program *traceroute* untuk melacak jalannya paket dari satu *host* ke *host* yang lain. Program *traceroute* dapat melakukan pelacakan rute berjalannya IP dengan cara

mengirimkan paket kecil UDP ke IP tujuan, dengan TTL yang diset membesar.

Saat paket pertama dikirim, TTL diset satu, sehingga *router* pertama akan membuang paket ini dan mengirimkan paket *ICMP time exceeded*. Kemudian paket ke-2 dikirim, dengan TTL dinaikkan. Dengan naiknya TTL, paket ini sukses melewati *router* pertama namun dibuang oleh *router* ke-2. *Router* ini pun mengirimkan paket *ICMP time exceeded*. Dengan mendaftar nama-nama *router* yang mengirimkan paket *ICMP time exceeded* ini, akhirnya didapat seluruh nama *router* yang dilewati oleh paket UDP.

- *Parameter Problem*. Paket ini dikirimkan jika terdapat kesalahan parameter pada *header* paket IP.
- *Source Quench*. Paket ICMP ini dikirimkan jika *router* atau tujuan mengalami kongesti. Sebagai respon atas paket ini, pihak pengirim paket harus memperlambat pengiriman paketnya.
- *Redirect*. Paket ini dikirimkan jika *router* merasa host mengirimkan paket IP melalui *router* yang salah. Paket ini seharusnya dikirimkan melalui *router* lain.

Sedangkan *ICMP Query Message* terdiri atas :

- *Echo* dan *Echo Reply*. Bertujuan untuk memeriksa apakah sistem tujuan dalam keadaan aktif. Program *ping* merupakan program pengirim paket ini. *Responder* harus mengembalikan data yang sama dengan data yang dikirimkan.
- *Timestamp* dan *Timestamp Reply*. Menghasilkan informasi waktu yang diperlukan sistem tujuan untuk memproses suatu paket.
- *Address Mask*. Untuk mengetahui berapa *netmask* yang harus digunakan oleh suatu *host* dalam suatu *network*.

Sebagai paket pengatur kelancaran jaringan, paket ICMP tidak diperbolehkan membebani *network*. Karenanya, paket ICMP tidak boleh dikirim saat terjadi problem yang disebabkan oleh :

- Kegagalan pengiriman paket ICMP.
- Kegagalan pengiriman paket *broadcast* atau *multicast*.

c. **ARP (Address Resolution Protocol)**

Dalam jaringan lokal, paket IP biasanya dikirim melalui *card ethernet*. Untuk berkomunikasi mengenali dan berkomunikasi dengan *ethernet* lainnya, digunakan *ethernet address*. *Ethernet address* ini besarnya 48 bit. Setiap *ethernet card* memiliki *ethernet address* yang berbeda-beda.

Pada saat hendak mengirimkan data ke komputer dengan IP tertentu, suatu *host* pada jaringan *ethernet* perlu mengetahui, di atas *ethernet address* yang manakah tempat IP tersebut terletak. Untuk keperluan pemetaan IP *address* dengan *ethernet address* ini, digunakan protocol ARP (*Address Resolution Protocol*).

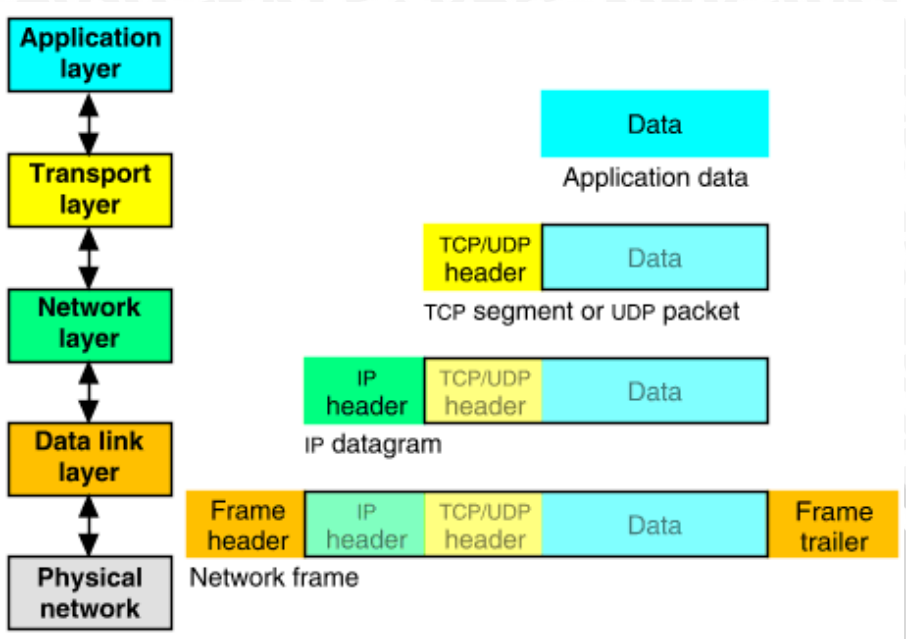
ARP bekerja dengan mengirimkan paket berisi IP *address* yang ingin diketahui alamat *ethernet*nya ke alamat *broadcast ethernet*. Karena dikirim ke alamat *broadcast*, semua *card ethernet* akan mendengar paket ini. *Host* yang merasa memiliki IP *address* ini akan membalas paket tersebut, dengan mengirimkan paket yang berisi pasangan IP *address* dan *ethernet address*. Untuk menghindari seringnya permintaan jawaban seperti ini, jawaban disimpan di memori (*ARP cache*) untuk sementara waktu.

2.3.1.4. *Network Interface Layer*

Layer ini bertanggungjawab mengirim dan menerima data dari media fisik. Contohnya ialah *Ethernet*.

2.3.2. Enkapsulasi Data

Enkapsulasi data adalah proses pembentukan dan pembungkusan data dalam format paket tertentu untuk setiap layer TCP/IP. Data dari lapisan aplikasi yang akan dikirimkan melalui jaringan akan mengalami proses enkapsulasi data pada setiap layer di bawahnya. Proses ini ditunjukkan dalam Gambar 2.5



Gambar 2.5 Proses Enkapsulasi Data

Sumber : [CAL-02]

Pada setiap lapisan yang dilalui ditambahkan sebuah *header* yang berisi informasi-informasi pengontrol komunikasi. Unit data yang akan diterima pada lapisan *transport* dibagi-bagi menjadi potongan data yang disebut segmen TCP atau *datagram* UDP untuk aplikasi yang menggunakan UDP sebagai protokol *transport*nya. Segmen tersebut diteruskan ke lapisan *network* dan disebut *datagram* IP. *Datagram* IP diteruskan pada lapisan *Data Link* untuk ditransmisikan melalui media fisik jaringan.

Jika suatu protokol menerima data dari protokol lain di *layer* atasnya, ia akan menambahkan informasi tambahan miliknya ke data tersebut. Informasi ini memiliki fungsi yang sesuai dengan fungsi protokol tersebut. Setelah itu, data ini diteruskan lagi ke protokol pada *layer* di bawahnya. Proses ini disebut *encapsulation.* Hal yang sebaliknya terjadi jika suatu protokol menerima data dari protokol lain yang berada pada *layer* di bawahnya. Jika data ini dianggap *valid*, protokol akan melepas informasi tambahan tersebut, untuk kemudian meneruskan data itu ke protokol lain yang berada pada *layer* di atasnya. Proses ini disebut *decapsulation.*

Setiap teknologi akses jaringan mempunyai ukuran maksimum *frame* data yang dapat ditransmisikan atau yang disebut dengan MTU (*Maximum Transmission Unit*). Bila ukuran datagram IP melebihi MTU maka IP memotong datagram menjadi

fragment-fragment yang mempunyai ukuran yang tidak melebihi MTU jaringan. Proses pemotongan datagram menjadi *fragment* disebut fragmentasi.

2.3.3. *Routing*

Routing berarti melewatkan paket IP ke *network* tujuan. Alat yang berfungsi melakukan *routing* paket ini disebut sebagai *router*. Agar mampu melewatkan paket data antar jaringan, maka *router* minimal harus memiliki dua buah *network interface*.

Proses *routing* dilakukan dengan *hop by hop* yang berarti paket data dilewatkan ke *router* terdekat hingga mencapai *network* tujuan. IP tidak mengetahui jalur keseluruhan menuju tujuan setiap paket. IP *routing* hanya menyediakan IP *address* dari *router* berikutnya (*next hop router*) yang menurutnya "lebih dekat" ke *host* tujuan.

Gambar 2.6 Proses *Routing*

Sumber : [ALM-01]

Paket data masuk ke dalam *router* melalui *input interface* kemudian *router* akan memproses (*forwarding*) sesuai dengan konfigurasi yang dimilikinya dan paket data yang telah diproses akan keluar melalui *output interface*. Proses *routing* teersebut, dijelaskan dalam Gambar 2.6.

Secara umum, *routing* dapat dibedakan menjadi dua, yaitu *static routing* dan *dynamic routing.*

2.3.3.1. *Static Routing*

Proses *routing* dengan metode *static routing* adalah dengan menuliskan jalur-jalur *routing* secara manual pada *router*. Jalur-jalur yang dituliskan tersebut, adalah jalur-jalur yang bersifat tetap mengingat *static routing* diperuntukkan untuk suatu jaringan bertopologi tetap. Dalam menggunakan metode ini, tidak boleh terdapat satu kesalahan pun karena setiap kesalahan dalam penulisan *routing* berarti kegagalan suatu jaringan untuk menghubungi jaringan lain.

2.3.3.2. *Dynamic Routing*

Dynamic routing adalah suatu protokol pada *router* yang dapat secara otomatis mencari jalur-jalur *routing* ke suatu tujuan tertentu di dalam suatu jaringan. Jika topologi suatu jaringan berubah maka protokol ini akan secara otomatis merubah

isi tabel *routing* dengan menyesuaikannya dengan topologi jaringan yang baru. Perubahan topologi jaringan ini dapat terjadi misalnya seperti adanya penambahan atau pengurangan *router* [MAL-02:5].

2.4. Topologi Jaringan

Setiap standar *Local Area Network* (LAN) memiliki peraturan pengkabelannya sendiri. Aturan ini mendefinisikan bagaimana menghubungkan media, perangkat keras yang dibutuhkan dan berbagai aturan pemasangan komponen. Dua hal utama berkaitan dengan media adalah tipe media (biasanya adalah tipe kabel) dan bagaimana berbagai kabel jaringan tersebut ditarik.

Hal penting yang harus dicatat bahwa terdapat perbedaan antara topologi jaringan secara fisik dan logika. Topologi fisik menjelaskan bagaimana media kabel dibentang, sedangkan topologi logika menjelaskan bagaimana perilaku jaringan. Token *ring* memiliki topologi *ring* karena data mengalir melalui setiap *station* sampai kembali ke *station* awal dimana data muncul, jadi data seperti melalui suatu *ring*. Tetapi secara fisik, jaringan token *ring* memiliki topologi *star* karena dibentuk dari kabel yang menghubungkan setiap *station* ke *hub*.

Skema pengaturan kabel dalam LAN disebut topologi pengkabelan LAN. Topologi yang umum dalam LAN adalah *Bus*, *Star* dan *Ring*.

2.4.1. Topologi *Bus*

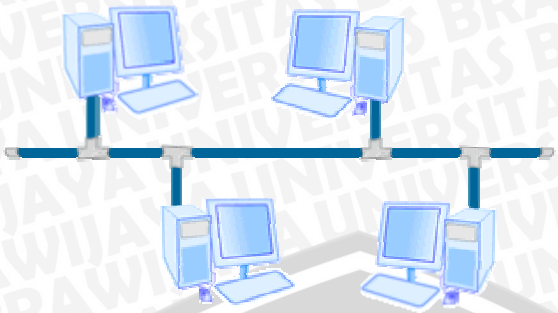
Topologi Bus terdiri dan kabel yang menghubungkan semua station. Sinyal yang dikirim dan suatu station akan terdistribusi ke semua station lain. Jadi karakteristik Topologi Bus adalah berbagi media (*shared media*). Keuntungan dari penggunaan topologi bus adalah:

- Hemat kabel.
- *Layout* kabel sederhana.
- Mudah dikembangkan.

Kekurangan penggunaan Topologi Bus adalah:

- Sulit untuk medeteksi kerusakan, apabila terjadi kerusakan dalam jaringan.
- Lalu lintas padat.
- Seluruh jaringan akan tidak berfungsi, jika terjadi kerusakan pada kebel utama.
- Diperlukan *repeater* untuk jarak jauh.

Contoh Topologi Bus ditunjukkan dalam Gambar 2.7.

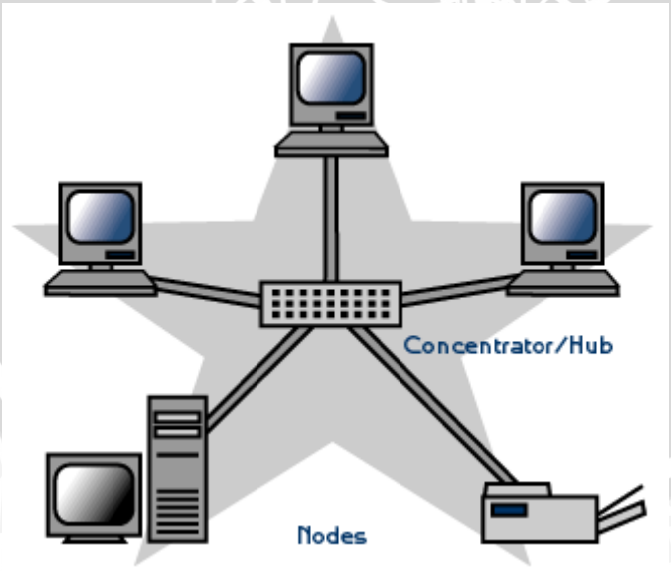


Gambar 2.7 Topologi Bus

Sumber : [CON-05a]

2.4.2. Topologi *Star*

Pada Topologi Star, setiap *station* terhubung melalui media kabel ke perangkat atau elemen saklar (switching element / hub) sentral. karena itu komunikasi antara dua *station* harus melalui perangkat tersebut. Jadi karakteristik topologi ini adalah tidak berbagi media. Perangkat saklar sentral rnempunyai kemampuan mengisolasi sinyal antar *port* sehingga kerusakan pada satu *port* tidak mempengaruhi *port* lainnya. Keuntungan lain adalah mudah dalam menghubungkan atau melepas hubungan *station* dan perangkat sentral. Contoh Topologi *Star* ditujukkan dalam Gambar 2.8.



Gambar 2.8 Topologi Star

Sumber : [FLO-05]

2.4.3. Topologi *Ring*

Topologi Ring terbentuk dan kabel yang membentuk lingkaran tertutup (loop) dengan station yang terhubung ke kabel tersebut. Sinyal hanya dikirim dalam satu arah aliran. Jadi karakteristik Topologi Ring adalah berbagi media. Keuntungan penggunaan Topologi Ring adalah:

- Mudah untuk mencari kerusakan dalam jaringan.
- Tidak terdapat *collision*.

Kekurangan dari penggunaan topologi ring adalah:

- Apabila satu komputer terganggu, komputer lainnya akan terganggu.
- Pengembangan jaringan lebih kaku

Contoh Topologi Ring ditujukkan dalam Gambar 2.9.

Gambar 2.9 Topologi Ring

Sumber : [CON-05b]

2.5. Iptables

Iptables adalah suatu program berbasis *command-line* yang berfungsi untuk membuat filter dan melakukan NAT (*Network Address Translation*) terhadap suatu paket data [NCT-07]. Iptables berjalan pada sistem operasi berbasis Linux yang memiliki kernel dengan kemampuan `ip_tables`, yaitu kernel versi 2.4.x dan 2.6.x. Iptables adalah suatu program *open source* yang berlisensi GPL.

Iptables terdiri dari banyak aturan yang tersusun dalam suatu struktur tertentu. Sebuah Tabel terdiri dari rantai-rantai yang memiliki cara pemrosesan paket data yang sama. Sedangkan Rantai adalah sekelompok aturan-aturan yang telah ditentukan. Iptables memiliki tiga buah rantai dasar, yaitu *INPUT, OUTPUT,* dan *FORWARD.*

Setiap aturan mengandung suatu parameter bagaimana suatu paket akan cocok dengannya, misalnya dengan nomor *port* atau nomor IP dan suatu keputusan jika paket tersebut cocok. Setiap paket data yang masuk ke dalam sistem akan dikirim ke suatu rantai untuk dicocokkan satu persatu dengan aturan-aturan yang ada di dalam rantai tersebut sesuai urutan. Jika paket data tersebut cocok dengan suatu aturan maka paket data tersebut akan ditindaklanjuti sesuai dengan kebijakan yang tertulis pada aturan tersebut. Jika paket data tersebut tidak cocok dengan suatu aturan apapun maka tindakan yang dilakukan adalah menjalankan kebijakan *default* dari Iptables.

2.5.1. Tabel

Iptables memiki tiga buah tabel yang berfungsi untuk menjalankan tiga hal yang berbeda. Tabel-tabel tersebut adalah sebagai berikut :

2.5.1.1. Tabel Filter

Tabel Filter berfungsi untuk melakukan penyaringan terhadap suatu paket data, apakah diijinkan untuk diproses atau tidak. Sebuah paket data akan melewati minimal satu dari tiga buah rantai yang dimiliki oleh Tabel Filter. Rantai-rantai tersebut adalah :

- Rantai *INPUT*

  Semua paket data yang akan memasuki sistem tempat Iptables berada, akan melalui rantai ini.
- Rantai *OUPUT*

  Semua paket data yang akan keluar dari sistem tempat Iptables berada, akan melalui rantai ini.
- Rantai *FORWARD*

  Semua paket data yang hanya akan melewati sistem tempat Iptables berada, akan melalui rantai ini.

2.5.1.2. Tabel NAT

Tabel NAT berfungsi untuk mengubah atau menulis ulang nomor IP atau *port* pada paket data. Tabel NAT memiliki tiga buah rantai yang bertugas memutuskan bagaimana semua paket data pada suatu koneksi akan ditulis ulang. Rantai-rantai tersebut adalah :

- Rantai *PREROUTING*

Paket data yang datang untuk masuk kedalam sistem akan melalui rantai ini sebelum diarahkan oleh tabel *routing* lokal. Rantai ini digunakan terutama untuk *destination-NAT* (DNAT).

- Rantai *POSTROUTING*

  Paket data yang keluar dari sistem akan melalui rantai ini setelah diarahkan oleh tabel *routing* lokal. Rantai ini digunakan terutama untuk *source-NAT* (SNAT).

- Rantai *OUTPUT*

  Rantai ini memperbolehkan DNAT terbatas pada paket yang dibuat pada sistem lokal.

2.5.1.3. Tabel MANGLE

Tabel *MANGLE* berfungsi untuk melakukan pengaturan pada *options* dari suatu paket data, misalnya *quality of service*. Semua paket data akan melalui tabel ini dan karena tabel ini didesain untuk mengatur suatu paket data dalam taraf ahli, maka tabel ini memiliki semua rantai-rantai yang ada. Rantai-rantai tersebut adalah :

- Rantai *PREROUTING*

  Paket data yang datang untuk masuk kedalam sistem, terlepas dari arah tujuan paket data apakah menuju ke sistem lokal atau ke sistem lain, akan melalui rantai ini sebelum diarahkan oleh tabel *routing* lokal.

- Rantai *INPUT*

  Semua paket data yang ditujukan kepada sistem lokal tempat Iptables berada, akan melewati tabel ini.

- Rantai *FORWARD*

  Semua paket data yang bertujuan ke sistem lain dengan melawati sistem lokal, akan melalui rantai ini.

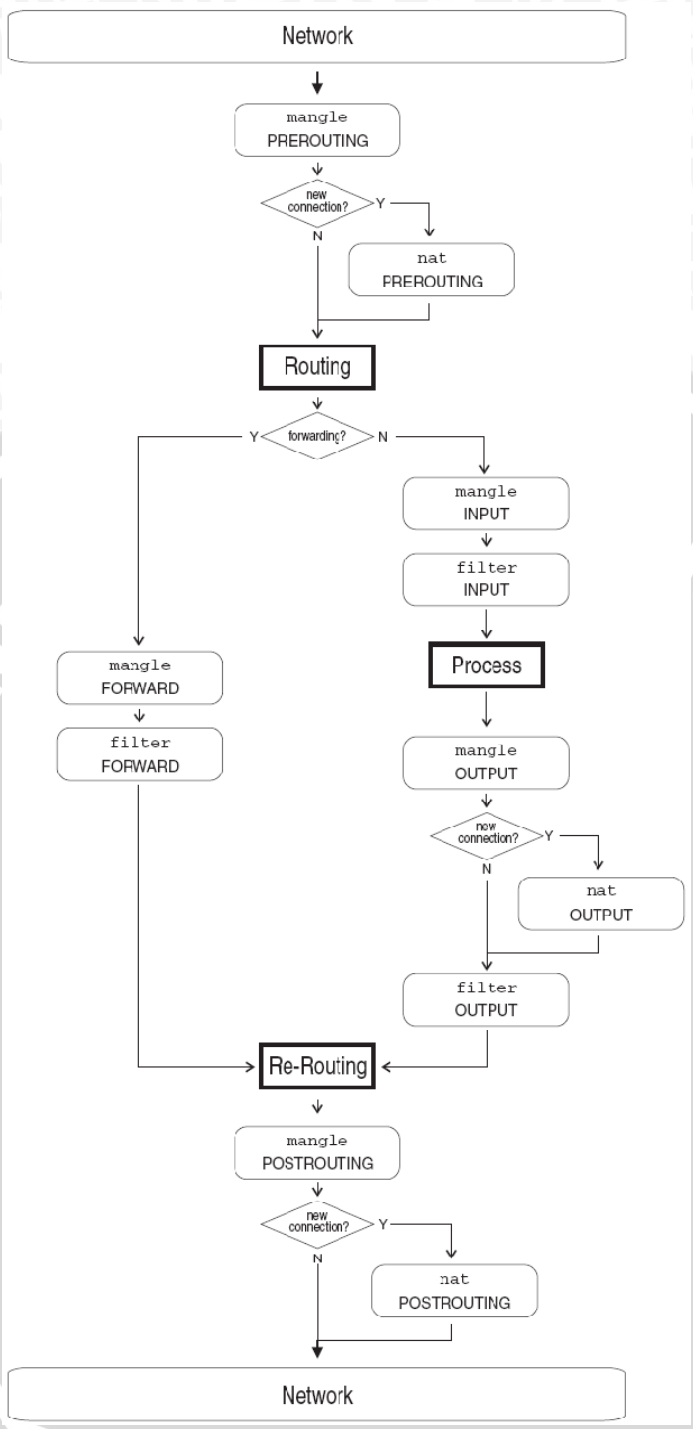
- Rantai *OUTPUT*

  Semua paket data yang keluar dari sistem lokal tempat Iptables berada, akan melewati tabel ini.

- Rantai *POSTROUTING*

  Semua paket data yang berasal dari sistem lain dan bertujuan ke sistem lain tapi melalui sistem lokal, akan melalui rantai ini.

Tabel-tabel yang telah disebutkan diatas, dapat lebih dijelaskan dalam Gambar 2.10.



Gambar 2.10 Diagram pada Iptables

Sumber : [WIK-06]

2.5.2. *Target* (Keputusan)

Target adalah suatu aturan yang berada dalam sebuah rantai yang bertugas memproses paket data jika paket data tersebut sesuai dengan parameter kecocokan

yang tertulis dalam rantai tersebut. *Target-target* yang dimiliki oleh Iptables adalah sebagai berikut :

- *ACCEPT*

  Perintah ini berfungsi mengijinkan paket data untuk melakukan tujuannya. Contohnya, suatu paket data di-*ACCEPT* pada rantai *INPUT*, maka paket data tersebut dapat melanjutkan perjalanannya menuju sistem lokal.

- *DROP*

  Perintah ini berfungsi untuk membuang paket data. Paket data yang dibuang tersebut akan hilang tanpa meninggalkan tanda-tanda atau peringatan apapun. Sistem pengirim paket data hanya akan mendapat pesan *requet time out* yang dapat diterjemahkan menjadi banyak hal.

- *QUEUE*

  Perintah ini berfungsai mengirimkan paket data untuk diantrikan di *user space*. Sebuah aplikasi yang memiliki *library* `libipq` dapat memproses paket ini. Jika tidak ada aplikasi yang memproses paket data tersebut, perintah *QUEUE* ini akan melakukan tindakan yang sama dengan *target DROP*.

- *RETURN*

  Perintah ini berfungsi untuk membuat paket berhenti melintasi aturan-aturan pada rantai tempat paket tersebut menemui *target RETURN*. Jika rantai tersebut merupakan *subchain* dari rantai yang lain, maka paket akan kembali ke *superset chain* di atasnya dan masuk ke baris aturan berikutnya. Apabila rantai tersebut adalah rantai utama, misalnya *INPUT*, maka paket akan dikembalikan kepada kebijakan *default* dari rantai tersebut.

- *REJECT*

  Perintah ini memiliki fungsi yang hampir sama dengan *DROP*. Jika *DROP* membuang paket data tanpa adanya peringatan dan pesan kepada pengirim paket data, maka *REJECT* akan mengirimkan pesan *error* kepada pengirim paket data. Perintah ini banyak dipakai dalam rantai *INPUT* atau *FORWARD* pada Tabel *Filter*.

- *LOG*

  Perintah ini berfungsi untuk melakukan *log* atau pencatatan terhadap suatu paket. Perintah ini dapat digunakan dalam rantai apapun dan sering digunakan untuk *debug* atau analisis kesalahan.

- ULOG

  Perintah ini berfungsi hampir sama dengan perintah *LOG*. Jika *LOG* megirimkan informasi ke *kernel log*, maka *ULOG* akan menyebarkan paket data ini melalui *netlink socket* sehingga program-program *userspace* yang terhubung dengan pada *socket* tersebut, akan menerima paket ini.

- DNAT

  Perintah ini berfungsi untuk mengubah nomor IP atau nomor *port* tujuan pada paket data dengan tujuan untuk *Network Address Translation*. Perintah ini hanya dapat digunakan pada rantai *OUTPUT* dan *PREROUTING* dalam tabel NAT.

- SNAT

  Perintah ini berfungsi untuk mengubah nomor IP atau nomor *port* sumber pada paket data dengan tujuan untuk *Network Address Translation*. Perintah ini hanya dapat digunakan pada rantai *POSTROUTING* dalam tabel NAT.

- *MASQUERADE*

  Perintah ini merupakan suatu bentuk terbatas dari *target* SNAT yang digunakan pada pengalamatan IP secara dinamis. Perintah ini banyak digunakan penyedia layanan internet yang menggunakan modem atau DSL sebagai penghubung.

2.6. Squid

Squid adalah suatu *proxy caching server* untuk *web client* yang memiliki kinerja tinggi, mendukung protokol-protokol *file transfer* antara lain FTP, HTTP, dan mampu menangani semua permintaan (*request*) dalam suatu proses tunggal (*single process*), *non-blocking*, dan dikendalikan oleh I/O [TEA-04].

Squid didesain untuk dapat bekerja pada sistem operasi Unix. Oleh karena itu, squid dapat bekerja pula di sistem operasi berbasis Unix, misalnya Linux. Squid dapat juga dijalankan pada sistem operasi Windows dengan bantuan program emulator. Hak cipta atas squid dimiliki oleh Universitas California San Diego dan squid berlisensi *GNU General Public License.*

Squid terdiri dari dari beberapa bagian, antara lain squid sebagai *server* utama, *dnsserver* sebagai *Domain Name System*, beberapa program tambahan seperti program untuk menulis ulang permintaan (*request)* dan melakukan autentifikasi, beberapa *tool* manajemen dan *client*.

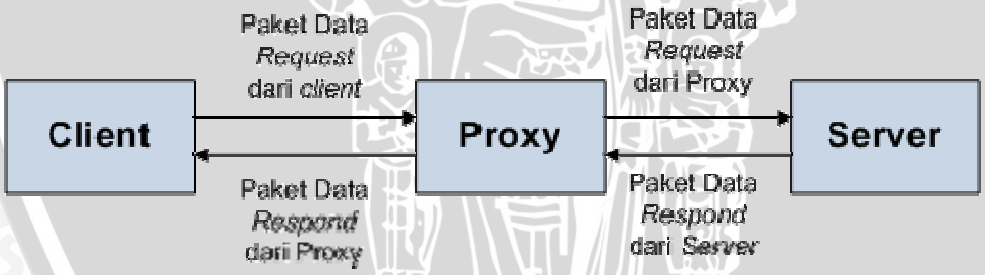
Fungsi utama dari squid adalah *web proxy chace* yang mana dapat didefinisikan sebagai suatu program yang menyimpan *web content* yang diminta

(*requested*) oleh *client* sehingga memudahkan *client-client* di *network* local dapat mengaksesnya dengan lebih cepat dan handal.

Kemampuan-kemampuan yang dimiliki squid antara lain adalah menahan *meta data* terutama objek-objek yang sering diakses tetap berada (*cached*) di memori melakukan *cache* pada *DNS lookups*, mendukung *non-blocking DNS lookups*, dan menerapkan *negative caching* pada *request* yang mengalami kegagalan. Squid juga mendukung *Secure Socket Layer*, kontrol akses lebih luas, dan dukungan penuh atas permintaan *log*.

Semua konfigurasi yang dibutuhkan oleh Squid untuk dapat bekerja, telah tertulis pada *syntax-syntax* yang tersimpan di dalam `squid.conf` dengan nilai *default*. Nilai-nilai pada *syntax* tersebut dapat diubah sesuai dengan kebutuhan.

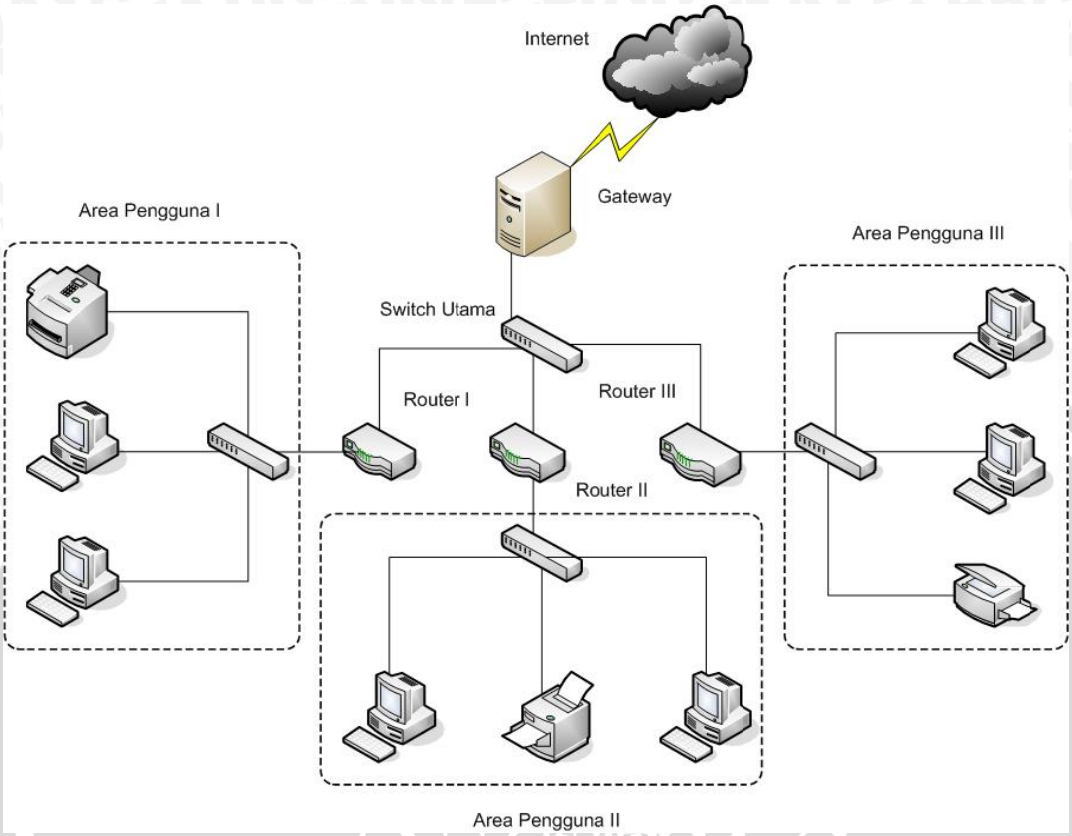
Secara umum, Squid melakukan proses *proxy* dengan cara menerima paket data *request* dari *client*. Selanjutnya Squid akan memeriksa data-data yang dimilikinya apakah ada yang sesuai dengan permintaan *client*, jika data tersebut tidak ada maka Squid akan melakukan *request* ke *server* tempat data yang diminta oleh *client* berada. Saat data dari *server* sampai pada Squid maka Squid akan memberikan data tersebut pada *client* yang memintanya dan menyimpan salinan data tersebut dalam dirinya [CHA-96]. Proses *proxy* diperlihatkan dalam Gambar 2.11 dibawah ini.



Gambar 2.11 Proses *Proxy*

Sumber : [CHA-96]

Gambar 2.12 dibawah ini adalah contoh gambar topologi jaringan tempat Squid akan diimplementasikan. Squid akan di-*install* pada komputer yang berfungsi sebagai *gateway.*

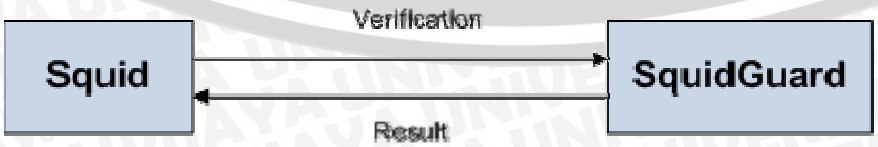


Gambar 2.12 Diagram Jaringan dengan *Gateway*

2.7. SquidGuard

SquidGuard adalah suatu program kombinasi antara *filter*, *redirector*, dan *access controller* yang merupakan suatu *plug-in* dari Squid. Karakteristik utama dari SquidGuard antara lain, sangat fleksibel, cepat, mudah proses instalasi-nya, *portable*, dan gratis karena berlisensi GNU [BAL-02].

Secara umum, cara kerja SquidGuard dapat dijelaskan dalam Gambar 2.13 dibawah ini.



Gambar 2.13 Cara Kerja SquidGuard

Sumber : [KOR-07]

Saat Squid akan meminta data ke *server* sesuai dengan permintaan *client*, Squid akan mengirim data tersebut kepada SquidGuard untuk diperiksa apakah sesuai dengan aturan yang dimiliki oleh SquidGuard. Selanjutnya, SquidGuard akan memberikan balasan kepada Squid tentang bagaimana tindakan atas data tersebut akan diberikan. Jika data sesuai dengan aturan maka data tersebut akan dikirim oleh Squid ke *server* tujuan, tapi jika tidak sesuai Squid akan mengirim suatu pesan tertentu sesuai dengan konfigurasi yang dimilikinya.

2.7.1. Fungsi

Fungsi-fungsi yang dimiliki oleh SquidGuard antara lain :

- Membatasi pengguna untuk mengakses suatu situs berdasarkan daftar *web server* atau *url* yang diijinkan.
- Memblokir pengaksesan ke *web server* atau url yang dilarang.
- Memblokir pengaksesan ke url-url yang memiliki ekspresi atau kata-kata tertentu yang dilarang.
- Me-*redirect* url yang diblokir ke suatu *url* yang lain.
- Me-*redirect* pengguna yang belum terdaftar ke halaman formulir pendaftaran.
- Me-*redirect* pen-*download*-an *file-file* yang sering di-*download* ke *local copies.*
- Me-*redirect* iklan-iklan ke suatu *file* GIF kosong.
- Dapat memiliki aturan akses berbeda berdasarkan hari, jam, tanggal, dan lain-lain.
- Dapat memiliki aturan yang berbeda untuk user yang berbeda.

2.7.2. Konfigurasi

Untuk dapat berfungsi sebagaimana yang diinginkan, SquidGuard perlu dikonfigurasi. Konfigurasi-konfigurasi ini dilakukan dengan menuliskan sintaks-sintaks pada *file* squidGuard.conf. Sintaks-sintaks berikut dapat dikombinasikan sesuai keperluan.

2.7.2.1. *Define Different Time Spaces*

Konfigurasi ini digunakan untuk menentukan ruang waktu. Sintaks-sintaksnya adalah sebagai berikut :

- *Time of Day*, misalnya (23:59 – 06:00, 09:00 – 15:00)
- *Day of the Week*, misalnya (*Saturday, Monday*)
- *Date*, misalnya (2007-07-07)

- *Date Range*, misalnya (2006-04-21 – 2006-05-05)
- *Date Wildcard*, misalnya (*-01-01 - *.03-07)

2.7.2.2. *Group Source*

Konfigurasi ini digunakan untuk menentukan user atau IP asal, misalnya direktur, manager, atau karyawan biasa. Sintaks-sintaksnya adalah sebagai berikut :

- *IP Address Range*, dengan :
  - Notasi *prefix*, misalnya 10.0.0.0/24
  - Notasi *netmask*, misalnya 10.0.0.0/255.255.255.0
  - Notasi *first-last*, misalnya 10.0.0.1 – 10.0.0.129
- *Address List*, misalnya 172.17.62.25, 172.17.63.253, dan lain-lain.
- *Domain List*, misalnya `mail.yahoo.com`, www.wikipedia.org, www.kaskus.us.
- *User ID List*, misalnya arioz, haunz, riekzt.

2.7.2.3. *Group Destination*

Konfigurasi ini digunakan untuk menentukan url, *server*, atau alamat tujuan. Sintaks-sintaks-nya adalah sebagai berikut :

- *Domain* (termasuk sub-domain)
- *Host*
- *Directory URLs* (termasuk *sub-directory*)
- *Regular Expression*

2.7.2.4. *Rewrite/Redirect URLs*

Konfigurasi ini digunakan untuk me-*redirect* atau meneruskan suatu url tertentu ke suatu url atau hal yang lain. Sintaks-sintaks-nya adalah sebagai berikut :

- *String/Regular Expression Editing*
- *URLs Replacement*

2.7.2.5. *Define Access Control List*

Konfigurasi ini digunakan mengatur akses melalui *control list.* Misalnya akses untuk membuka site-site yang diperbolehkan atau akses untuk membuka site-site yang tidak diijinkan, yang telah tercantum dalam daftar kontrol (*control list).*

2.7.2.6. *Logging*

Konfigurasi ini digunakan untuk menulis *log* atas koneksi-koneksi atau kelompok-kelompok yang diinginkan.

2.8. Protokol HTTP (*Hyper Text Transfer Protocol*)

Protocol HTTP adalah suatu cara untuk melakukan pertukaran dan penyampaian data melalui *world wide web*. Fungsi asli dari protocol ini adalah untuk mempublikasikan dan mendapatkan suatu halaman HTML (*Hyper Text Markup Language*) (*Anonymous*, 2006) [INT-99].

Pengembangan protokol HTTP dikoordinasi oleh *World Wide Web Consortium* dan *Internet Engneering Task Force* yang akan merumuskan suatu RFC (*Request for Comments*) yang digunakan sebagai standar dunia.

HTTP adalah suatu protokol yang berada diantara *client* dan *server*. *Client*, yang biasanya berupa *web browser* disebut sebagai *user agent*. Sedangkan *server* yang dituju, yaitu pihak yang menyediakan *web content* seperti halaman html atau gambar-gambar disebut sebagai *origin server*. Diantara *user agent* dan *origin server* mungkin terdapat beberapa perantara, misalnya *proxy* atau *gateway*.

Sebuah *client* HTTP mengajukan suatu permintaan (*request*) atas *web content* dengan cara membuat suatu koneksi TCP dengan suatu *port* (biasanya *random port* diatas 1024) pada sebuah *remote host*. Sedangkan sebuah *server* HTTP akan mendengarkan (*listening*) *port* 80 dan menunggu apakah ada suatu permintaan *web content* dari *client*. Setelah menerima suatu permintaan dari *client*, *server* akan membalas dengan mengirimkan suatu informasi status seperti "HTTP/1.1 200 OK" dan informasi lain sesuai yang dimiliki *server* tersebut, misalnya halaman yang diminta oleh *client*, pesan *error*, atau pesan-pesan yang lainnya.

Untuk mendapat akses ke suatu *web content* yang menggunakan protokol HTTP, maka digunakan suatu *url* tertentu, yaitu `http:`

2.8.1. Pesan *Request*

Pesan *request* berisi hal-hal berikut :

- *Request Line*, misalnya `GET /images/naruto_logo.gif HTTP/1.1`, yang merupakan suatu permintaan file `naruto_logo.gif` dari direktori `/images`.
- *Headers*, misalnya `Accept-Language: id`
- Sebuah baris kosong
- Sebuah *optional messege body*

Sebuah baris *request* dan *headers*, semua harus berakhiran dengan CRLF (*Carriage Return* yang diikuti dengan *Line Feed*). Baris kosong harus mengandung CRLF saja

dan tidak mengandung *whitespace* lain. Beberapa *headers* hanyalah pilihan, sedangkan yang lain bisa saja harus karena dibutuhkan oleh protokol HTTP/1.1.

2.8.2. Metode *Request*

Protokol HTTP memiliki delapan metode yang menunjukkan hal yang ingin dilakukan pada resources yang diketahui. Metode-metode tersebut adalah :

- *HEAD* berfungsi untuk meminta *response identical* yang cocok dengan *GET request*, akan tetapi tanpa *response body*.
- *GET* berfungsi untuk meminta "perwakilan" dari *resource* tertentu.
- *POST* berfungsi untuk memasukkan data yang akan diproses (misalnya formulir html) ke dalam *resource* yang diketahui.
- *PUT* berfungsi untuk melakukan *upload* "perwakilan" dari *resource* tertentu.
- *DELETE* berfungsi menghapus *resource* tertentu.
- *TRACE* berfungsi untuk mengirim pesan konfirmasi bahwa sebuah permintaan telah diterima sehingga *client* dapat melihat apa yang telah ditambahkan atau diubah oleh *intermediate servers*.
- *OPTIONS* berfungsi untuk mengetahui metode-metode HTTP yang didukung oleh *server* sehingga dapat digunakan untuk memeriksa fungsionalitas *web server*.
- *CONNECT* berfungsi untuk mengubah ke SSL (*Secure Socket Layer) tunnel* jika digunakan dengan *proxy*.

HTTP server dibuat dengan minimal memiliki setidaknya metode *GET* dan *HEAD* dan jika memungkinkan, juga terdapat metode *OPTIONS*.

2.8.3. Metode *Safe*

Metode-metode yang dikategorikan aman adalah metode yang hanya digunakan untuk melakukan pencarian informasi dan tidak mengubah informasi apapun yang berada di *server*. Metode-metode yang dapat dikategorikan sebagai *safe* adalah *GET* dan *HEAD*.

Metode seperti *GET* secara umum adalah aman, tapi patut diwaspadai bahwa metode ini juga dapat menimbulkan suatu efek samping misalnya penghapusan suatu *record* di *database* dengan hanya suatu *hyperlink* sederhana yang terdapat pada halaman HTML.

2.8.4. Metode *Idempotent*

Metode-metode seperti *GET, HEAD, PUT,* dan *DELETE* dapat disebut dengan *idempotent* yang berarti bahwa suatu permintaan (*request*) sama yang dilakukan berkali-kali akan menghasilkan efek yang sama dengan suatu permintaan yang dilakukan hanya sekali.

2.8.5. Versi HTTP

Protokol HTTP telah banyak dikembangkan menjadi beberapa versi dan umumnya, versi-versi tersebut mendukung versi sebelumnya atau *backward-compatible*. Secara umum, saat *client* menghubungi *server* untuk pertama kali, *client* akan memberitahu *server* versi HTTP yang digunakannya. Selanjutnya, *server* akan membalas dengan menggunakan versi yang sama pula.

2.8.5.1. 0.9

Versi ini hanya mendukung satu metode yaitu *GET* dan tidak mendukung *headers* dan *POST*. Karena versi ini tidak mendukung *POST* maka *client* tidak dapat terlalu banyak memberikan informasi kepada *server*.

2.8.5.2. HTTP/1.0

Versi ini adalah revisi pertama yang menetapkan versi protokol dalam suatu komunikasi dan sampai saat ini masih tetap digunakan secara luas, terutama oleh *proxy server*.

2.8.5.3. HTTP/1.1

Versi ini adalah versi yang digunakan saat ini. Versi ini secara *default* telah mendukung *persistent connections* dan dapat bekerja dengan baik pada *proxy*, mendukung *request pipelining*, dapat mengirimkan banyak permintaan (*request*) dalam satu waktu, memberikan cukup waktu bagi *server* untuk mengadapi beban kerja (*workload*) dan memungkinkan *server* untuk membalas permintaan *client* dengan lebih cepat.

2.8.5.4. HTTP/1.2

Versi ini pertama kali muncul pada tahun 1995 oleh W3C dan hingga saat ini masih dalam tahap pengembangan.

2.8.6. Kode Status

Pada HTTP versi 1.0 keatas, baris pertama pada *HTTP response* disebut *status line. Status line* tersebut mengandung kode status dan *reason phrase*. Kode status

merupakan kode angka tertentu misalnya 403, sedangkan *reason phrase* berisi suatu informasi misalnya *Forbidden*. Sebuah respon yang akan ditangani oleh *user agent* dengan suatu cara berdasarkan kode sebagai pertimbangan utama dan *response header* sebagai pertimbangan kedua. Jika *user agent* mendapat suatu kode yang tidak dikenali, maka *user agent* tersebut dapat memberikan kode status tersendiri (*custom*) dengan *digit* pertama dari kode tersebut digunakan untuk menentukan respon umum dari kode tersebut.

Pada *reason phrase*, informasi standar yang terdapat di dalamnya hanyalah sebuah rekomendasi dan dapat diubah dengan isi yang lain tergantung dari kebijakan dari pembuat *web*. Jika kode status menunjukkan adanya suatu masalah, maka *user agent* dapat menampilkan *reason phrase* yang berisi tentang informasi lebih lanjut tentang bagaimana menangani masalah tersebut. Selanjutnya, sesuai dengan standar yang ada, *user agent* juga dapat langsung mencoba untuk menerjemahkan atau melakukan tindakan berdasarkan isi dari *reason phrase*. Walaupun hal ini sebaiknya tidak dilakukan mengingat secara jelas disebutkana bahwa kode status adalah *machine-readable* (dapat dipahami oleh mesin) dan *reason phrase* adalah *human-readable* (dapat dipahami manusia).

2.8.7. *Persistent Connections*

Pada protokol HTTP versi 0.9, suatu koneksi ditutup setelah sepasang *request-response* terjadi. Pada versi 1.0 hal tersebut juga hal standar, tetapi pada awal kerja dari HTTP versi 1.1, banyak implementasi yang dimulai dengan memberikan mekanisme *"keep-alive"* yang memungkinkan suatu koneksi dapat digunakan lebih dari satu kali. Mekanisme ini dapat juga disebut *persistent connections* karena mampu mengurangi *lag* (ketertinggalan). Hal ini disebabkan karena *client* tidak perlu lagi melakukan negosiasi ulang pada koneksi TCP setelah permintaan (*request*) pertama dikirim.

Jika pada HTTP/1.0 diperlukan suatu *header* khusus *"Keep-Alive"* untuk menggunakan *persistent connections*, pada HTTP/1.1 hal itu tidak diperlukan karena telah menjadi *default*. Versi 1.1 juga memperkenalkan c*hunked encoding* yang dapat membuat isi (*content*) dari *persistent connections* menjadi ter-*stream* bukan ter-*buffer* dan HTTP *Pipelining* yang membuat *client* dapat mengirimkan beberapa tipe permintaan (*request*) sebelum respon dari permintaan sebelumnya diterima. Hal tersebut juga dapat mengurangi *lag*.

2.8.8. HTTP *Session state*

Protokol HTTP bersifat *stateless* yang artinya bahwa suatu permintaan (*request*) bersifat mandiri, tidak berhubungan dengan permintaan apapun sebelumnya. Untuk mempertahankan *users's state* pada suatu *web*, misalnya saat suatu *host* ingin melakukan pengaturan *content* untuk pengguna yang pernah mengunjungi mereka, maka salah satu metode yang umum digunakan adalah dengan mengirimkan dan meminta *cookies.*

2.8.9. *Secure* HTTP

Saat ini terdapat dua metode untuk membangun suatu koneksi HTTP *secure*, yaitu skema https URI dan header HTTP 1.1 *Upgrade*. Metode yang paling umum digunakan diantara kedua metode tersebut adalah skema https URI, sedangkan penggunaan header HTTP 1.1 *Upgrade* hampir tidak pernah digunakan.

BAB III
METODELOGI PENULISAN

Tahap ini akan menjelaskan langkah-langkah yang akan dilakukan untuk merancang sistem yang akan dibuat hingga dapat berfungsi sebagaimana yang diharapkan. Langkah-langkah tersebut adalah sebagai berikut :

3.1. Studi Literatur

Studi literatur yang dilakukan bertujuan untuk mengkaji hal-hal yang berhubungan dengan teori-teori yang mendukung dalam perencanaan dan perancangan sistem, yaitu :

1. Kajian pustaka mengenai Protokol TCP/IP, yaitu suatu protokol yang digunakan oleh suatu komputer dalam suatu jaringan komputer agar dapat saling berkomunikasi dengan komputer lain.
2. Kajian pustaka mengenai Topologi Jaringan Komputer, yaitu suatu cara untuk menyusun komputer-komputer secara fisik hingga membentuk sebuah jaringan komputer
3. Kajian pustaka mengenai Squid, yaitu suatu server *web-cache proxy* yang merupakan program induk untuk SquidGuard.
4. Kajian pustaka mengenai SquidGuard, yaitu suatu program *plug-in* bagi Squid yang berfungsi untuk melakukan pemblokiran dan pe-*redirect*-an url.

3.2. Perancangan dan Implementasi Sistem

Pada tahap ini, sistem untuk memfilter layanan internet pada *gateway* dengan menggunakan SquidGuard dirancang sebagaimana berikut :

1. Perancangan topologi jaringan komputer
2. Pemasangan (*install*) sistem operasi pada setiap komputer di jaringan komputer.
3. Pemasangan (*install*) Squid dan SquidGuard pada komputer *gateway*.
4. Pengkonfigurasian Squid dan SquidGuard pada komputer *gateway* sehingga dapat bekerja sesuai dengan yang diharapkan.

3.3. Pengujian dan Analisis Sistem

Pada tahap pengujian dan analisis sistem ini, SquidGuard yang telah dibangun diuji dengan beberapa cara untuk menembus filter yang telah dibuat. Hasil dari pengujian tersebut akan dianalisis, apakah sistem yang dibangun telah memenuhi harapan atau belum.

3.4. Pengambilan Kesimpulan dan Saran

Tahap ini adalah tahap pengambilan kesimpulan dari sistem yang telah dibuat. Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan dan pengujian sistem telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan ini merupakan informasi akhir dari perancangan sistem yang berisi mengenai berhasil atau tidaknya sistem tersebut dijalankan.

Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi serta menyempurnakan penulisan.

BAB IV
PERANCANGAN SISTEM

Perancangan sistem pemfilteran layanan internet pada *gateway* dengan menggunakan SquidGuard ini dibagi menjadi dua tahap utama, yaitu perancangan perangkat keras dan perancangan perangkat lunak. Perancangan perangkat keras adalah merancang topologi jaringan komputer. Perancangan perangkat lunak meliputi: perancangan jaringan komputer secara *logical*, perancangan *gateway*, perancangan sistem pemfilter layanan internet.

4.1. Analisis kebutuhan

Analisis kebutuhan dalam perancangan Sistem Pemfilteran layanan Internet pada Gateway dengan menggunakan SquidGuard terdiri dari Spesifikasi Sistem dan *Analisis Context Diagram* dan *Data Flow Diagram* (DFD).

4.1.1. Spesifikasi Sistem

Spesifikasi Sistem Pemfilteran layanan Internet pada Gateway dengan menggunakan SquidGuard adalah sebagai berikut:

a. Terdapat dua *network*, yaitu *internal network* dan *external network*.
b. *Internal network* adalah *network* tempat sistem akan diimplementasikan. *Network* ini meliputi sejumlah *client* dan sebuah *gateway* yang dihubungkan dengan Topologi *Star* dan memiliki pengenal (*network ID*) 10.100.100.0/25. Penggunaan Topologi *Star* pada *network* ini dimaksudkan agar semua aliran paket data yang akan keluar dari *internal network* akan melewati *gateway*. Penggunaan *network* ID 10.100.100.0/25 dimaksudkan agar penggunaan IP di dalam *internal network* benar-benar terpisah dari *network* lain.
c. *External network* adalah *network* lain yang terhubung ke *internal network*. *Network* ini meliputi *network* Universitas Brawijaya dan Internet. *Network* Universitas Brawijaya melingkupi *internal network* dan memiliki pengenal (*network* ID) 172.17.63.129/25.
d. Semua perangkat lunak yang dirancang pada bab ini, akan diimplementasikan pada komputer *gateway* dengan pertimbangan bahwa topologi jaringan komputer akan lebih sederhana dan perangkat keras yang digunakan lebih sedikit.

e. Sistem Pemfilteran yang dirancang pada bab ini disusun oleh Squid yang diimplementasikan pada *Gateway* dan Squid menggunakan SquidGuard sebagai *plug-in.*

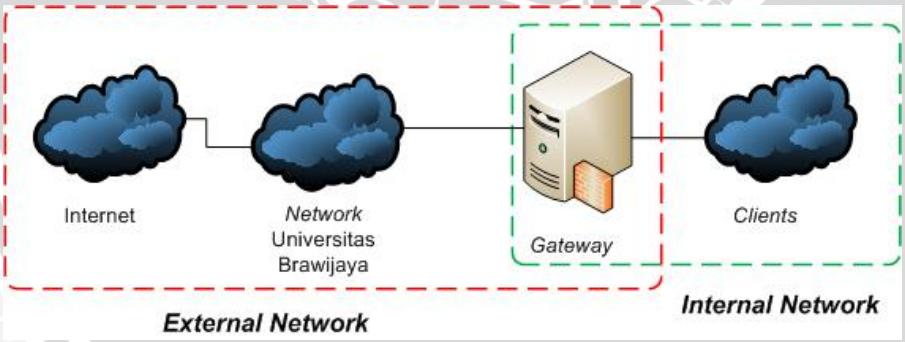
f. Squid berfungsi sebagai program induk bagi SquidGuard dan bukan sebagai *web cache.*

g. SquidGuard digunakan untuk memblokir akses ke suatu situs berkategori dewasa berdasarkan daftar *web server* atau *url* yang diijinkan pada saat jam kerja, yaitu Hari Senin sampai Jum'at pukul 07.00 hingga 15.30.

h. SquidGuard digunakan untuk memblokir akses *file-file* multimedia seperti wma, avi, mp3, dan 3gp berdasarkan kata-kata atau ekspresi yang terdapat di *url* pada saat jam kerja, yaitu Hari Senin sampai Jum'at pukul 07.00 hingga 15.30.

i. SquidGuard digunakan untuk mengarahkan *url* yang diblokir ke suatu halaman yang menunjukkan bahwa *url* yang diakses tersebut untuk tidak dapat diakses pada saat jam kerja dan dalam 7 detik, halaman tersebut akan *refresh* ke www.brawijaya.ac.id

Dalam Gambar 4.1, diperlihatkan diagram jaringan yang menunjukkan *internal network* dan *external network.*



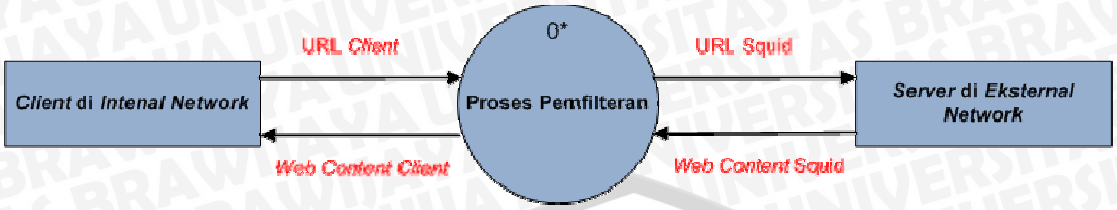
Gambar 4.1 Diagram Jaringan *Internal Network* dan *External Network*

4.1.2. Analisis *Context Diagram* dan *Data Flow Diagram* (DFD)

Context Diagram merupakan diagram yang menampilkan masukan proses, proses dan keluaran proses dari sistem secara umum. *Context Diagram* adalah DFD yang pertama kali dibuat dikenal dengan DFD level 0.

DFD level 0 *Client* ke *Server* merupakan diagram yang menjelaskan secara umum tentang masukan proses, proses, dan keluaran proses yang terjadi pada *client* di

internal network yang menuju ke *server* di *external network.* DFD level 0 *Client* ke S*erver* ditunjukkan dalam Gambar 4.2.



Gambar 4.2 DFD Level 0 *Client* ke *Server*

Berdasarkan Gambar 4.2, Sistem Pemfilteran memiliki kamus data sebagai berikut :

- URL *Client*

  URL *Client* merupakan paket data yang dikirim oleh *browser* milik *client* di dalam *internal network* menuju suatu *server* yang berada di *external network.*

- URL Squid

  URL Squid merupakan paket data milik Squid yang mewakili *client* untuk melakukan *request* ke *server* tujuan

- *Web Content* Squid

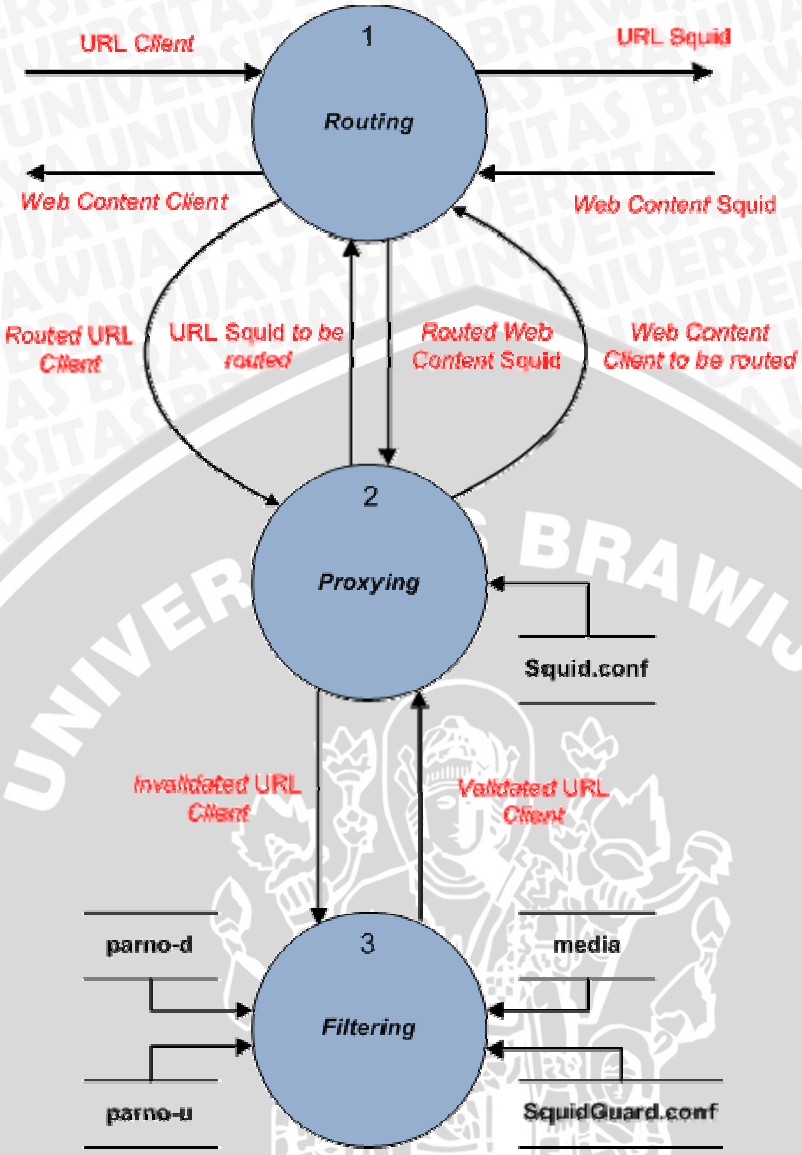
  Web Content Squid merupakan paket data balasan dari *server* di *external network* atas paket data URL Squid.

- *Web Content Client*

  Web Content Client merupakan paket data dari Squid sebagai balasan atas *request* yang diterima Squid dari *client* yang meminta *web content* dari suatu *server* di *external network.*

Proses Pemfilteran yang ditunjukkan dalam Gambar 4.2 merupakan suatu proses yang terbentuk sebagai hasil dari konfigurasi dari Sistem Pemfilteran. URL *Client,* URL Squid, *Web Content* Squid, *Web Content Client* merupakan data yang berbentuk *raw bit* yang bergerak melalui media transmisi.

DFD level 1 *Client* ke S*erver* merupakan penjelasan yang lebih detail dari DFD level 0 *Client* ke S*erver*. DFD level 1 *Client* ke S*erver* ditunjukkan dalam Gambar 4.3.



Gambar 4.3 DFD level 1 *Client* ke *Server*

Sebuah *client* yang berada di *internal network* mengirimkan suatu paket data informasi URL ke sebuah *server* di *external network*. Sebelum mencapai *server*, paket data informasi URL tersebut melewati sebuah *gateway*. Di dalam *gateway*, paket data informasi URL akan diproses oleh Squid. Proses tersebut bernama proses *proxy*. Didalam proses *proxy*, Squid akan melakukan proses verifikasi paket data URL dengan bantuan SquidGuard yang mana proses ini disebut proses *filter*. Hasil dari proses *filter* akan digunakan oleh Squid untuk memutuskan bagaimana tindakan yang akan dilakukan atas paket data informasi URL. Jika paket data informasi URL sesuai dengan aturan-aturan yang dimiliki oleh SquidGuard maka Squid akan meneruskannya ke *gateway* untuk kemudian di-*routing* ke *server* tujuan. Jika paket data informasi URL tidak sesuai dengan aturan-aturan yang dimilki oleh SquidGuard

maka Squid akan mengirimkan pesan sesuai dengan konfigurasi yang dimilikinya ke *gateway* untuk di-*routing*-kan ke *client* asal.

Balasan yang diberikan oleh *server* atas paket data informasi URL, akan melewati *gateway*. Di dalam *gateway*, paket data balasan dari *server* akan diproses oleh Squid yang disebut proses *proxy* kemudian paket data tersebut di-*routing* ke *client* tujuan.

Saat Squid dijalankan, Squid akan membaca konfigurasi pada *file* squid.conf dan akan mengaktifkan SquidGuard. Saat SquidGuard dijalankan, SquidGuard akan membaca konfigurasi pada *file* squidGuard.conf, data *domain* situs berkategori dewasa di *file* parno-d, data *subdomain* situs berkategori dewasa di *file* parno-u dan data *file-file* multimedia di *file* media.

DFD level 1 *Client* ke *Server* memiliki kamus data yang dijelaskan sebagaimana berikut :

- *Routed* URL *Client*

  Routed URL *Client* merupakan data URL yang melewati *gateway* kemudian ditangkap dan dibaca oleh Squid.

- *Invalidated* URL *Client*

  Invalidated URL *Client* merupakan URL yang dikirim oleh Squid ke SquidGuard untuk bandingkan dengan konfigurasi yang dimiliki oleh SquidGuard.

- *Validated* URL *Client*

  Validated URL *Client* merupakan URL yang diteruskan oleh SquidGuard ke Squid setelah melalui proses pemfilteran.

- URL Squid *to be routed*

  URL Squid *to be routed* merupakan URL dari Squid yang mewakili *client* untuk melakukan *request* ke *server* di *external network* sesuai dengan *request* dari *client.*

- *Routed Web Content* Squid

  Routed Web Content Squid merupakan *web content* yang diterima oleh Squid sesuai dengan permintaan dari *client.*

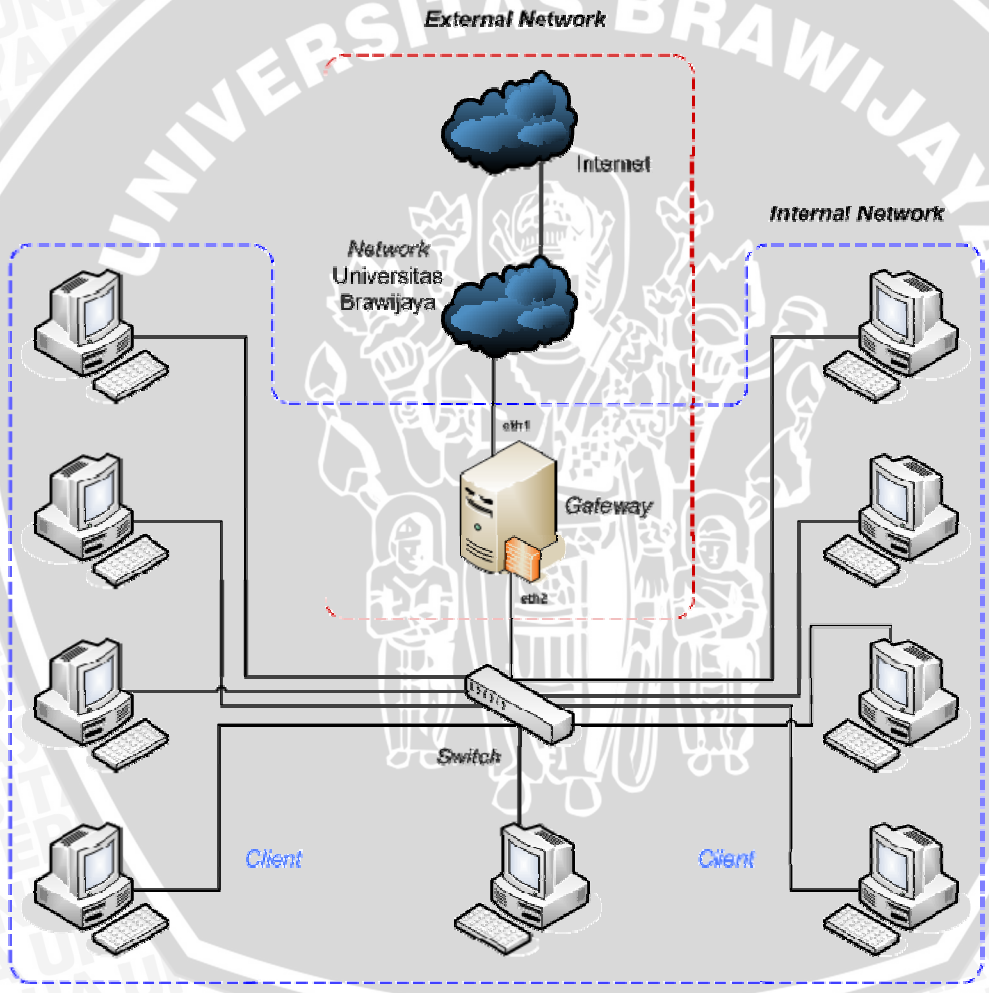
- *Web Content Client to be routed*

  Web Content Client to be routed merupakan *web content* yang dikirim oleh Squid sebagai balasan kepada *client* sesuai permintaan.

Proses Pemfilteran yang ditunjukkan dalam Gambar 4.3 merupakan suatu proses yang terbentuk sebagai hasil dari konfigurasi *Gateway*, Squid, dan SquidGuard. Semua data yang dijelaskan dalam kamus data DFD level 1 *Client* ke *Server* merupakan data berbentuk URL.

4.2. Perancangan Perangkat Keras

Perancangan perangkat keras adalah merancang suatu topologi jaringan komputer tempat sistem yang akan diimplementasikan. Topologi jaringan komputer yang digunakan pada sistem ini adalah Topologi Star, sebagaimana yang diperlihatkan dalam Gambar 4.4.



Gambar 4.4 Diagram Jaringan Tempat Implementasi Sistem

Dalam Gambar 4.2 di atas, dapat dijelaskan bahwa komputer *client* sebanyak 9 buah terhubung ke sebuah *switch.* Selanjutnya *switch* tersebut terhubung ke Internet melalui komputer *gateway*.

Perangkat lunak yang akan dirancang, seluruhnya akan diimplementasikan di komputer *gateway*.

4.3. Perancangan Perangkat Lunak

Perangkat lunak yang akan diimplementasikan dirancang dengan beberapa tahap secara berurutan, yaitu : perancangan jaringan komputer secara *logical*, perancangan *gateway*, dan perancangan sistem pemfilteran.

4.3.1. Perancangan Jaringan Komputer Secara *Logical*

Topologi jaringan komputer yang secara fisik telah dirancang tersebut, perlu pula dirancang secara *logical*. Perancangan secara *logical* adalah perancangan jaringan komputer dengan cara melakukan konfigurasi pada perangkat lunak dari perangkat-perangkat yang terhubung di jaringan komputer sehingga perangkat-perangkat tersebut dapat saling berkomunikasi. Konfigurasi tersebut adalah melakukan pengalamatan dengan IP *Address.*

Network yang digunakan dalam topologi ini adalah 10.100.100.0/25 yang berarti terdapat 126 buah nomor IP *host*, sebuah nomor IP *network*, dan sebuah nomor IP *broadcast*. Komputer *gateway* akan diberi nomor IP : 10.100.100.63 dengan netmask 255.255.255.128, sedangkan komputer *client* akan diberi konfigurasi sebagai berikut :

- *Address Type* : *Manually Configured*
- IP *Address* : 10.100.100.3 sampai dengan 10.100.100.10 dan 10.100.100.12
- *Subnet Mask* : 255.255.255.128
- *Default Gateway* : 10.100.100.63
- *Primary* DNS : 202.162.208.99
- *Secondary* DNS : 202.162.208.100

4.3.2. Perancangan *Gateway*

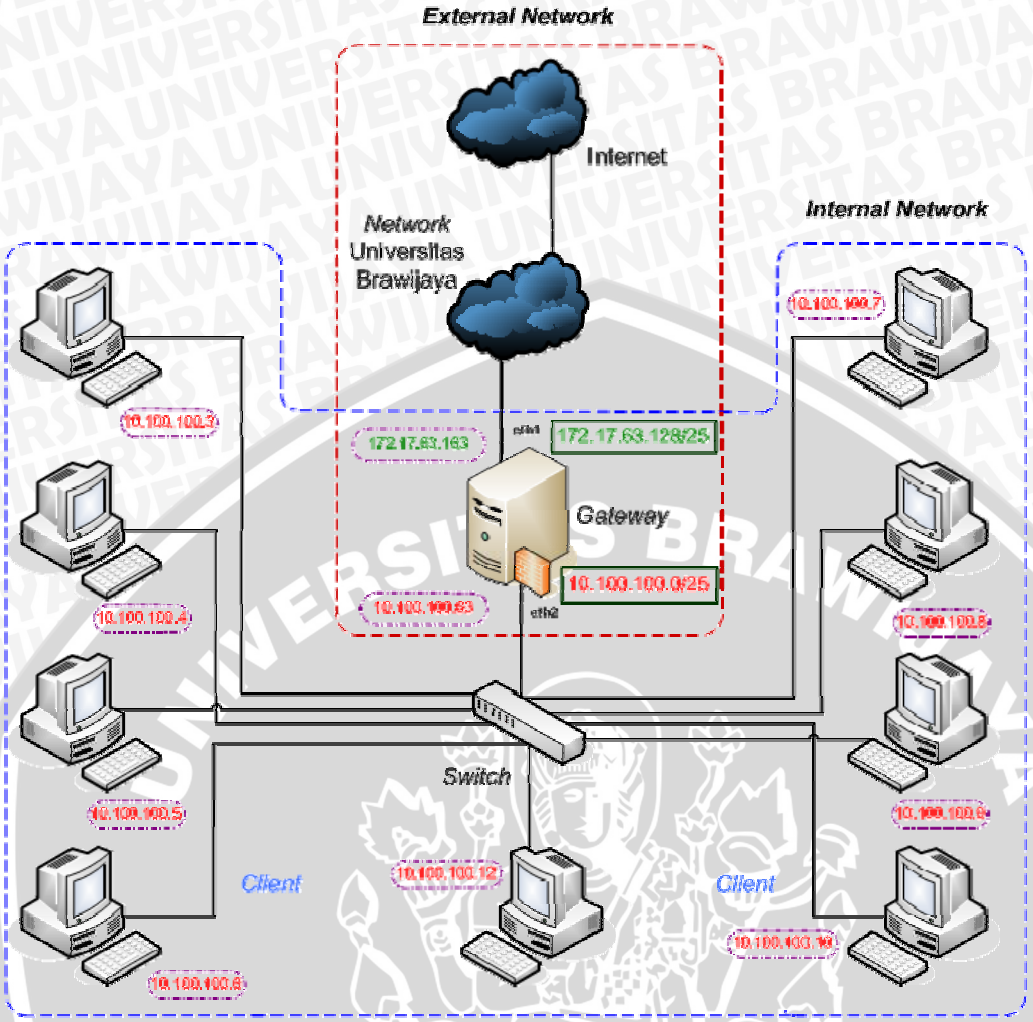
Komputer *gateway* adalah komputer yang berfungsi sebagai pintu masuk utama bagi suatu paket data saat memasuki *network* tertentu. *Gateway* sebenarnya adalah sebuah *router* yang secara *logical* menghubungkan *network* dari sisi dalam (*intranet*) ke *network* yang berada di sisi luar (*internet*). Komputer *gateway* yang

dirancang ini merupakan *end router*, yaitu suatu *router* yang langsung terhubung dengan *client* atau *host*, bukan dengan dengan *router* lain.

Setelah Sistem Operasi Fedora Core 6 di-*install*, maka agar komputer *gateway* berfungsi sebagai *gateway*, perlu dikonfigurasi sebagai berikut :

a. Mengaktifkan fungsi untuk meneruskan paket data
b. Memasang *Default Gateway* : 172.17.63.129
c. Memasang *Primary* DNS : 202.162.208.99 dan *Secondary* DNS : 202.162.208.100
d. Pada eth1, sebagai *uplink*, dikonfigurasi sebagai berikut :
   - BootProto : *None*
   - IP *Address* : 172.17.63.163
   - *Subnet Mask* : 255.255.255.128
   - *Default Gateway* : 172.17.63.129
e. Pada eth2, sebagai *downlink*, dikonfigurasi sebagai berikut :
   - BootProto : *None*
   - IP *Address* : 10.100.100.63
   - *Subnet Mask* : 255.255.255.128

Setelah melakukan konfigurasi pada jaringan komputer secara *logical* dan melakukan konfigurasi pada komputer *gateway*, maka topologi jaringan komputer akan tampak sebagaimana dalam Gambar 4.5.



Gambar 4.5 Diagram Jaringan Setelah Konfigurasi

4.3.3. Perancangan Sistem Pemfilteran

Sistem pemfilteran yang dirancang adalah sistem yang dibangun dengan menggunakan *software* Squid dan SquidGuard sebagai *plug-in*-nya pada Sistem Operasi Fedora Core 6.

4.3.3.1. Konfigurasi Squid

Untuk mendapatkan Squid yang berfungsi secara tepat sebagai program induk bagi SquidGuard dan bukan sebagai *web cache*, maka perlu dilakukan beberapa konfigurasi tambahan selain konfigurasi yang secara *default* telah dimiliki oleh Squid. Konfigurasi tersebut adalah sebagai berikut :

a. Membuka *port* 3128
b. Mengaktifkan *Transparent Proxy.*
c. Menentukan besar memori yang digunakan untuk melakukan *cache* sebesar 32 MB

d. Menentukan besar minimal *web content* yang akan di-*cache* sebesar 100 MB
e. Menentukan besar maksimal *web content* yang akan di-*cache* sebesar 100.001 KB
f. Menentukan direktori tempat *file cache* akan disimpan.
g. Menentukan SquidGuard sebagai *plug-in* bagi Squid.
h. Membuka akses bagi *network* 172.17.63.128/25 dan 10.100.100.0/25 untuk dapat mengakses Squid

4.3.3.2. Konfigurasi SquidGuard

Pemberian konfigurasi pada SquidGuard dilakukan sesuai dengan latar belakang penulisan Tugas Akhir ini yaitu pembatasan layanan akses Internet sesuai dengan ketentuan yang dibuat oleh instansi. Pembatasan tersebut diterapkan dengan cara memblokir situs-situs tertentu, pembatasan akses pada waktu-waktu tertentu, atau kombinasi dari ketiga hal tersebut.

Pemblokiran terhadap suatu situs dapat dijelaskan sebagai berikut : paket data dari *client* yang berisi *request* ke suatu situs akan memasuki komputer *gateway* melalui eth2. Paket data ini kemudian akan dibaca dan dibandingkan dengan konfigurasi yang dimiliki SquidGuard. Konfigurasi di SquidGuard ini meliputi sumber paket data, tujuan paket data, pemilik paket data, waktu paket data diterima, dan *access control* paket data. SquidGuard akan memutuskan bagaimana paket data akan diproses selanjutnya. Jika paket data tersebut memenuhi aturan yang terdapat pada konfigurasi SquidGuard, maka paket tersebut akan diteruskan ke eth1 untuk melanjutkan perjalanannya ke situs tujuan. Jika paket data tersebut tidak memenuhi aturan yang terdapat pada konfigurasi SquidGuard, maka paket tersebut akan dibuang.

Konfigurasi yang dilakukan pada SquidGuard adalah sebagai berikut :

a. Menetapkan waktu pembatasan akses layanan Internet, yaitu hari Senin sampai Jum'at pukul 07.30 hingga pukul 15.30
b. Situs berkategori dewasa tidak dapat diakses pada jam kerja. Contoh situs tersebut antara lain :
   - duniasex.com → Semua *domain* yang mengandung kata "duniasex.com" seperti : http://duniasex.com, ftp://duniasex.com, apa.duniasex.com, terserah.duniasex.com akan diblokir.

- film.dodol.sch.id/porn → Semua *subdomain* pada *domain* film.dodol.sch.id yang mengandung kata "*porn*" seperti : film.dodol.sch.id/porn, film.dodol.sch.id/hai-hai/porn, film.dodol.sch.id/porn/matamu akan diblokir.

c. *File* multimedia tidak dapat di-*download* pada saat jam kerja. Contoh *file-file* multimedia antara lain : *.avi, *.mp3, *.wma, *.3gp.

BAB V
IMPLEMENTASI SISTEM

Sistem pemfilteran layanan Internet pada *gateway* dengan menggunakan SquidGuard yang telah dirancang pada Bab Empat, akan diimplementasikan pada tahap ini. Ada dua tahap utama yang dilakukan pada bagian implementasi sistem ini, yaitu implementasi jaringan komputer dan implementasi sistem untuk memfilter layanan Internet.

5.1. Implementasi Jaringan Komputer

Pada tahap ini, rancangan jaringan komputer yang telah dibuat pada bab sebelumnya akan diwujudkan. Jaringan komputer terdiri dari dua bagian besar, yaitu bagian perangkat keras dan bagian perangkat lunak.

5.1.1. Perangkat Keras Jaringan Komputer

Pada bagian perangkat keras, komputer yang akan berfungsi sebagai *gateway* memiliki spesifikasi sebagai berikut :

- *Processor* : Intel Pentium 4 - 1.50 GHz
- *Motherboard* : ECS P4VMM2
- Harddisk : Maxtor 30 GB - 5400 RPM
- *Memory* : Visipro 256 MB PC 2700
- LAN *Card* :
  - Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+
  - D-Link System Inc RTL8139 Ethernet
- *Graphic Card* : Eagle 64 MB nVidia RIVA TNT2

Sedangkan komputer yang berfungsi sebagai *client*, memiliki spesifikasi yang bervariasi, tetapi cukup untuk melakukan koneksi Internet. Perangkat keras lainnya adalah sebagai berikut :

- *Switch* : *Switch Unmanageable* D-Link DES-1024R 24 *port*
- Kabel UTP : kabel **BELDEN**CDTNETWORKING™
- Konektor : RJ-45 AMP buatan Tyco Corporation

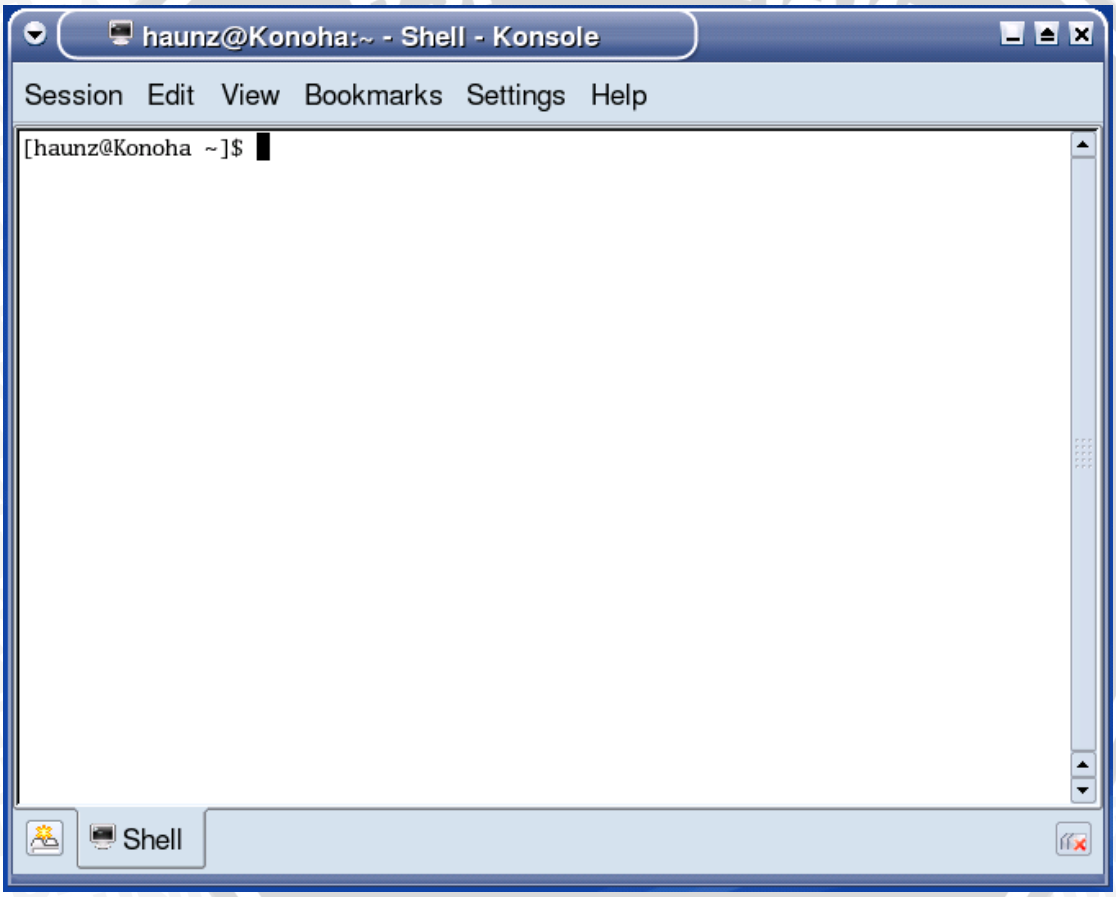
Perangkat keras tersebut kemudian disusun sehingga membentuk sebuah jaringan komputer dengan Topologi *Star* seperti yang diperlihatkan dalam Gambar 4.1.

5.1.2. Perangkat Lunak Jaringan Komputer

Pada tahap ini, perangkat keras yang telah disusun, dikonfigurasi perangkat lunaknya. Perangkat keras yang dikonfigurasi adalah komputer *gateway* dan komputer *client*

5.1.2.1. Konfigurasi Komputer *Gateway*

Konfigurasi pada komputer *gateway* dilakukan setelah Sistem Operasi Fedora Core 6 ter-*install*. Semua konfigurasi yang berupa perintah-perintah linux, dilakukan pada *Terminal – Command Line*. Sebuah *Terminal – Command Line* yang terdapat pada Fedora Core 6 dengan GUI (*Graphical User Interface*) KDE versi 3.5 diperlihatkan dalam Gambar 5.1



Gambar 5.1 ***Terminal – Command Line***

Untuk dapat melakukan konfigurasi, maka perlu *login* sebagai root. Untuk berpindah dari *user* biasa menjadi root, perintahnya adalah :

```
[haunz@Konoha ~]$ su -
Password:
[root@Konoha ~]#
```

Konfigurasi yang perlu dilakukan agar komputer *gateway* dapat berfungsi dengan baik adalah sebagai berikut :

a. Mengaktifkan fungsi untuk meneruskan paket data

Konfigurasi ini dilakukan dengan melakukan perubahan variabel dari 0 menjadi 1 pada parameter net.ipv4.ip_forward yang terdapat dalam *file* sysctl.conf

```
[root@Konoha ~]# vim /etc/sysctl.conf
```

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

b. Memasang *Default Gateway*

Konfigurasi ini dilakukan dengan menambahkan baris GATEWAY=172.17.63.129 dalam *file* network

```
[root@Konoha ~]# vim /etc/sysconfig/network
```

```
NETWORKING=yes
HOSTNAME=Konoha
GATEWAY=172.17.63.129
```

c. Memasang DNS

Konfigurasi ini dilakukan dengan menambahkan baris nameserver 202.162.208.99 dan nameserver 202.162.208.100 dalam *file* resolv.conf

```
[root@Konoha ~]# vim /etc/resolv.conf
```

```
search lasif.net
nameserver 202.162.208.99
nameserver 202.162.208.100
```

d. Konfigurasi eth1 sebagai *uplink*

Konfigurasi ini dilakukan dengan melakukan perubahan pada *file* ifcfg-eth1 seperti dibawah ini :

```
[root@Konoha ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth1
```

```
DEVICE=eth1
BOOTPROTO=static
BROADCAST=172.17.63.255
IPADDR=172.17.63.163
NETMASK=255.255.255.128
GATEWAY=172.17.63.129
```

```
NETWORK=172.17.63.128
ONBOOT=yes
```

e. Konfigurasi eth2 sebagai *downlink*

Konfigurasi ini dilakukan dengan melakukan perubahan pada *file* ifcfg-eth2 seperti dibawah ini :

```
[root@Konoha ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth2
```

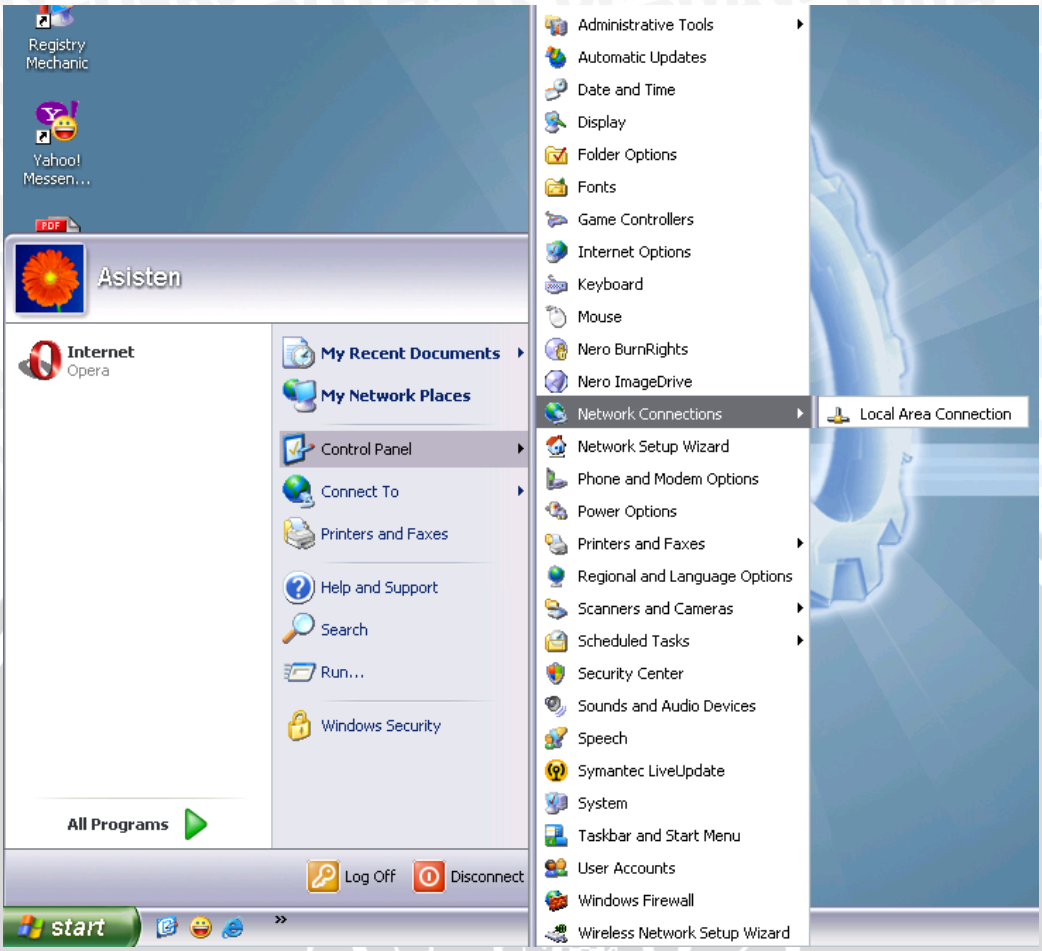
```
DEVICE=eth2
BOOTPROTO=static
BROADCAST=10.100.100.127
IPADDR=10.100.100.63
NETMASK=255.255.255.128
NETWORK=10.100.100.0
ONBOOT=yes
```

5.1.2.2. Konfigurasi Komputer Client

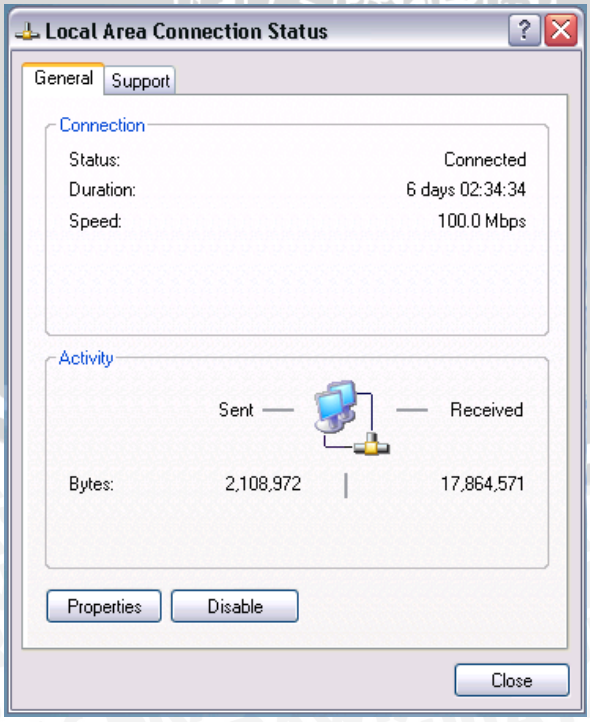
Konfigurasi pada komputer *client* dilakukan agar komputer tersebut dapat berkomunikasi dengan komputer *gateway*. Konfigurasi yang dilakukan adalah dengan memberikan nomor IP pada setiap komputer *client*.

Contoh cara konfigurasi pada komputer *client* yang menggunakan Sistem Operasi Microsoft Windows XP Professional Edition Service Pack 2 adalah sebagai berikut :

a. Pilih menu *Local Area Connection* pada *Control Panel*

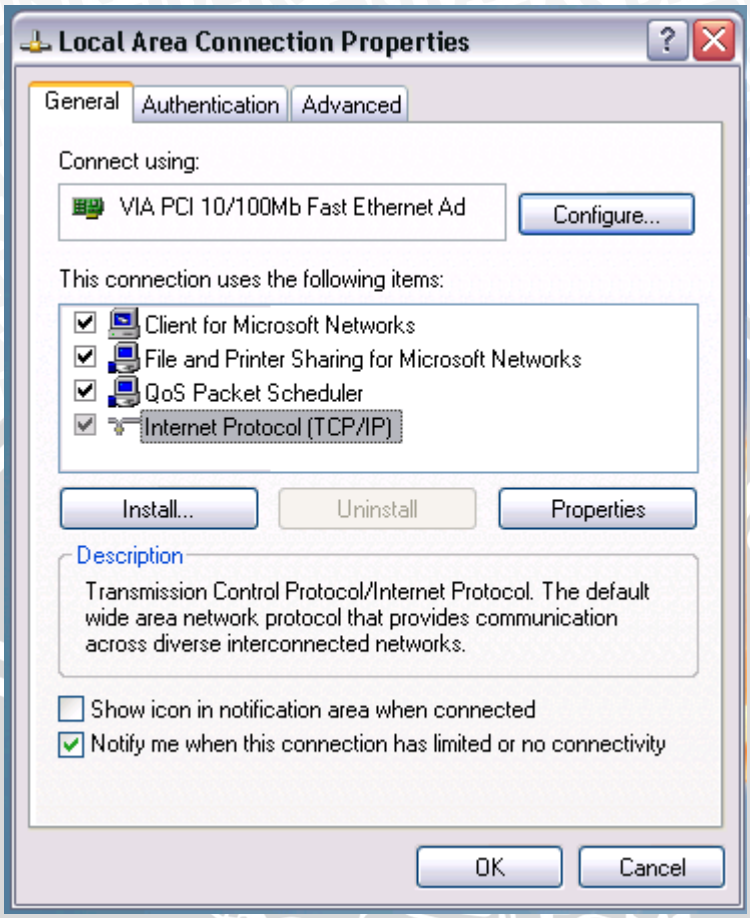


Gambar 5.2 *Local Area Connection* di *Control Panel*



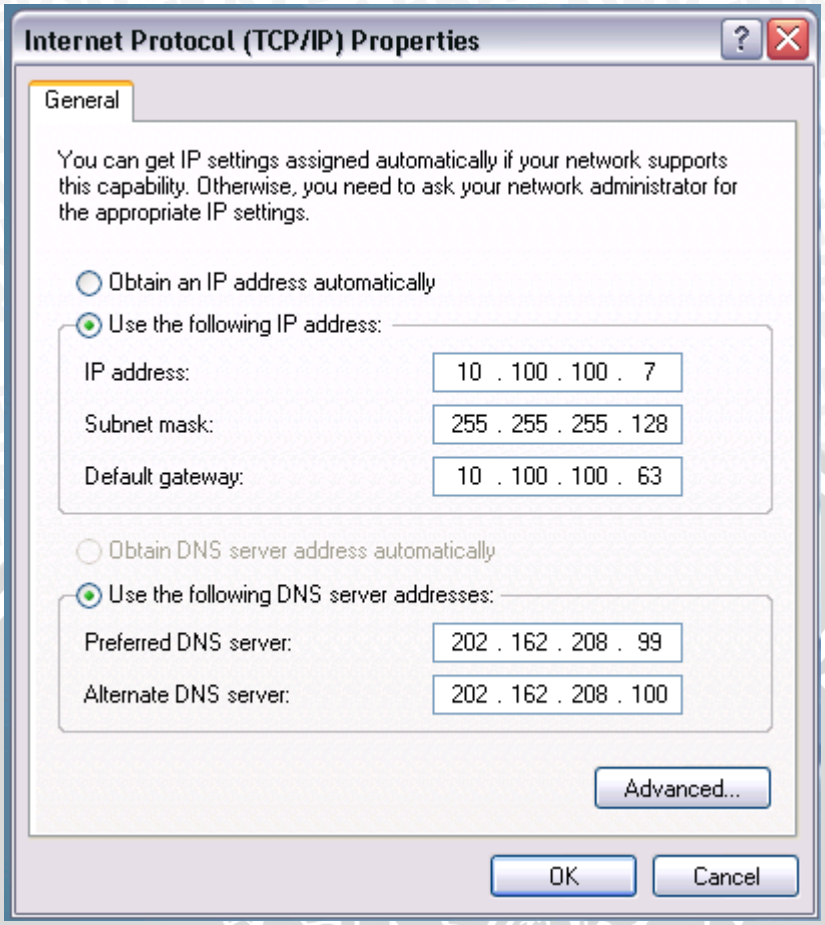
Gambar 5.3 *Local Area Connection Status*

b. Sesuai dalam Gambar 5.3, pada *tab – General* pilih *Properties.* Selanjutnya akan muncul *window* sebagaimana tampak dalam Gambar 5.4



Gambar 5.4 ***Local Area Connection Properties***

c. Sesuai dalam Gambar 5.4, pada *tab – General* pilih klik dua kali pada bagian *Internet Protocol* (TCP/IP). *Window* yang muncul, diisi dengan nilai sesuai dengan yang tertera dalam Gambar 5.5



Gambar 5.5 *Window* Pengisian Nomor IP

Jika pengisian telah selesai dilakukan, maka klik satu kali pada tombol OK untuk menyimpan konfigurasi.

Contoh selanjutnya adalah konfigurasi komputer *client* yang menggunakan Sistem Operasi Linux yang berbasis Red Hat Linux, misalnya Linux CentOS 4.3. Konfigurasinya adalah sebagai berikut :

a. Konfigurasi dilakukan pada *Terminal – Command Line* dan *login* sebagai root.

Untuk berpindah dari *user* biasa menjadi root, perintahnya adalah :

```
[haunz@Rasengan ~]$ su -
Password:
[root@Rasengan ~]#
```

b. Memasang *Default Gateway*

Konfigurasi ini dilakukan dengan menambahkan baris GATEWAY=10.100.100.63 dalam *file* network

```
[root@Rasengan ~]# vim /etc/sysconfig/network
```

```
NETWORKING=yes
HOSTNAME=Rasengan
GATEWAY=10.100.100.63
```

c. Memasang DNS

Konfigurasi ini dilakukan dengan menambahkan baris nameserver 202.162.208.99 dan nameserver 202.162.208.100 dalam *file* resolv.conf

```
[root@Rasengan ~]# vim /etc/resolv.conf
```

```
search lasif.net
nameserver 202.162.208.99
nameserver 202.162.208.100
```

d. Konfigurasi eth0

Konfigurasi ini dilakukan dengan melakukan perubahan pada *file* ifcfg-eth0 seperti di bawah ini :

```
[root@Rasengan ~]# vim /etc/sysconfig/network-
scripts/ifcfg-eth0
```

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=10.100.100.127
HWADDR= 00:13:8F:1C:DD:74
IPADDR=10.100.100.3
NETMASK=255.255.255.128
GATEWAY=10.100.100.63
NETWORK=10.100.100.0
ONBOOT=yes
```

5.2. Implementasi Sistem pada *Gateway*

Pada tahap ini, rancangan sistem yang telah dibuat diimplementasikan pada komputer *gateway* yang telah dikonfigurasi pada tahap sebelumnya. Implementasi sistem yang dibuat meliputi dua bagian utama yaitu, implementasi Squid dan implementasi SquidGuard.

5.2.1. Implementasi Squid

Squid merupakan suatu *proxy server* yang secara *default* telah terdapat paket instalasi Linux Fedora Core 6. Oleh karena itu, dalam tahap ini implementasi Squid adalah melakukan konfigurasi pada Squid yang telah ter-*install* pada sistem. Squid yang secara *default* ter-*install* pada sistem adalah Squid versi 2.6

```
[root@Konoha ~]# rpm -qa | grep squid
squid-2.6.STABLE4-1.fc6
```

Agar Squid dapat berfungsi secara tepat sebagai program induk bagi SquidGuard dan bukan sebagai *proxy*, maka perlu dilakukan beberapa konfigurasi tambahan selain konfigurasi yang secara *default* telah dimiliki oleh Squid. Konfigurasi tersebut dilakukan dengan cara menambahkan beberapa baris *syntax* sesuai dengan kebutuhan pada *file* squid.conf. Penambahan baris baru sebaiknya diletakkan dibawah baris tempat *syntax* asli berada.

```
[root@Konoha ~]# vim /etc/squid/squid.conf
```

Konfigurasi-konfigurasi yang ditambahkan adalah sebagai berikut :

a. Membuka *port* 3128

Port 3128 adalah *unknown port* yang secara *default* digunakan oleh Squid saat menjalankan fungsinya.

```
http_port 3128
```

b. Menentukan besar memori yang digunakan untuk melakukan *cache* sebesar 32 MB

```
cache_mem 32 MB
```

Konfigurasi ini adalah untuk menentukan besar ukuran ideal *memory* yang digunakan oleh Squid untuk melakukan *cache* pada *In-Transit Objects*, *Hot Objects*, dan *Negative-Cached Objects*.

c. Menentukan besar minimal *web content* yang akan di-*cache* sebesar 100 MB

```
minimum_object_size 102400 KB
```

Konfigurasi ini adalah untuk menentukan besar minimal ukuran dari *web content* yang akan di-*cache*. *Web content* yang berukuran kurang dari 102.400 KB tidak akan di-*cache*. Ukuran minimal *web content* memang jauh di atas besar ukuran *memory* yang digunakan untuk melakukan *cache*, agar Squid tidak melakukan *cache web content*.

d. Menentukan besar maksimal *web content* yang akan di-*cache* sebesar 102.401 KB

```
maximum_object_size 102401 KB
```

Konfigurasi ini adalah untuk menentukan besar maksimal ukuran dari *web content* yang akan di-*cache*. *Web content* yang berukuran lebih dari 102.401 KB tidak akan di-*cache*. Besar ukuran minimal dan ukuran maksimal dari *web content* hanya selisih satu byte saja, agar kemungkinan Squid untuk melakukan *cache web content* menjadi sangat kecil.

e. Menentukan direktori tempat *file cache* akan disimpan.

```
cache_dir ufs /var/spool/squid 50 16 256
```

Konfigurasi ini adalah untuk menentukan lokasi tempat Squid menyimpan *web content*. Parameter `ufs` adalah *format* penyimpanan yang digunakan Squid, parameter `50` berarti besar ruang harddisk yang digunakan dalam *directory* `/var/spool/squid` adalah 50 MB, parameter `16` berarti jumlah *subdirectory* Tingkat Pertama (*Level* 1) dari *directory* `/var/spool/squid` adalah 16 buah, dan parameter `256` berarti jumlah *subdirectory* Tingkat Kedua (*Level* 2) yang akan dibuat dalam setiap *subdirectory* Tingkat Pertama (*Level* 1) adalah 256 buah.

f. Menentukan SquidGuard sebagai *plug-in* bagi Squid.

```
url_rewrite_program /usr/bin/squidGuard -c
/etc/squid/squidGuard.conf
```

Konfigurasi ini adalah agar Squid menggunakan SquidGuard sebagai *plug-in*.

g. Menentukan banyak proses dari *rewrite program* yang dapat bekerja secara paralel

```
url_rewrite_children 25
```

Konfigurasi ini menentukan Squid untuk menciptakan 25 buah proses secara paralel dari *rewrite program*. Jika jumlah proses tersebut terlalu sedikit, akan memperlambat kinerja Squid dalam menangani *request*. Sedangkan jika terlalu banyak, akan memperlambat kerja sistem lain.

h. Membuka akses bagi *network* 172.17.63.128/25 dan 10.100.100.0/25 untuk dapat mengakses Squid.

- Pada bagian `acl` (*Access Control List*), tambahkan :

```
acl net1 src 172.17.63.128/25
acl net2 src 10.100.100.0/25
```

Konfigurasi di atas adalah untuk mendefinisikan `net1` dan `net2`

- pada bagian `http_access`, tambahkan :

```
http_access allow net1
http_access allow net2
```

Konfigurasi diatas adalah untuk mengijinkan `net1`, yaitu *network* 172.17.63.128/25 dan `net2`, yaitu *network* 10.100.100.0/25 berkomunikasi dengan protokol HTTP

i. Memberi nama *hostname* pada Squid.

```
visible_hostname Konohagakure-no-Proxy
```

Untuk mengaktifkan *Transparent Proxy* pada *Squid*, maka konfigurasi yang perlu ditambahkan adalah sebagai berikut :

a. Menambahkan *syntax* transparent di akhir baris http_port 3128 pada *file* squid.conf

```
http_port 3128 transparent
```

Konfigurasi di atas adalah untuk mengaktifkan fungsi *transparent* pada Squid.

b. Menambahkan aturan pada IPTABLES untuk membelokkan semua paket data yang menuju *port* 80 dan 33444 ke *port* 3128

```
[root@Konoha ~]$ vim /etc/sysconfig/iptables
```

```
-A PREROUTING -p tcp -i eth2 -s 10.100.100.0/25
     -dport 80 -j REDIRECT -to--ports 3128
-A PREROUTING -p tcp -i eth2 -s 10.100.100.0/25
     -dport 33444 -j REDIRECT -to--ports 3128
```

5.2.2. Implementasi SquidGuard

Implementasi SquidGuard dilakukan setelah Squid dapat berfungsi dengan tepat untuk menjalankan SquidGuard dan bukan sebagai *proxy*. Implementasi SquidGuard terdiri dari dua tahap, yaitu tahap *installation* dan tahap konfigurasi.

5.2.2.1. Tahap *Installation*

SquidGuard merupakan suatu program *plug-in* bagi Squid dan tidak terdapat dalam paket *installation* Linux Fedora Core 6. Oleh karena itu, SquidGuard perlu di-*install* untuk dapat menjalankannya pada Squid. Proses *installation* SquidGuard adalah sebagai berikut :

```
[haunz@Konoha ~]$ su -
Password:
[root@Konoha ~]# cd /home/haunz/installer
[root@Konoha installer]# ls -lh
```

```
-rw------- 1 haunz haunz 2.3M Jan 11 19:53
    squidGuard-1.2.0-14.fc6.i386.rpm
[root@Konoha installer]# yum install squidGuard-
1.2.0-14.fc6.i386.rpm
```

5.2.2.2. Tahap Konfigurasi

Setelah tahap *installation* selesai, SquidGuard dapat langsung dikonfigurasi agar dapat bekerja. Seluruh konfigurasi SquidGuard terdapat di dalam *file* squidGuard.conf yang berada di `/etc/squid`. Konfigurasi-konfigurasi yang ditambahkan adalah sebagai berikut :

```
[root@Konoha ~]# vim /etc/squid/squidGuard.conf
```

a. Menentukan letak *directory* tempat SquidGuard membaca *file database*

```
dbhome /var/squidGuard/cream
```

b. Menentukan letak *directory* tempat SquidGuard menulis *file log*

```
logdir /var/log/squid
```

c. Menentukan rentang waktu

```
time bebas {
     weekly mtwhf 00:00 - 07:29
     weekly mtwhf 15:31 - 24:00
     date   *.01.01
     date   *.08.17
}
```

Rentang waktu yang ditentukan untuk SquidGuard tidak melakukan proses pemfilteran adalah sebagai berikut :

- Senin – Jum'at, pukul 15:31 hingga 07.29
- Tahun Baru Masehi setiap tanggal 1 Januari
- Hari Kemerdekaan RI setiap tanggal 17 Agustus

d. Menentukan sumber paket data yang akan diproses

```
src lasif {
    ip      10.100.100.0/25
}
```

Konfigurasi ini menentukan bahwa semua paket data yang berasal dari *network* 10.100.100.0/25 akan diproses oleh SquidGuard.

e. Merumuskan tujuan paket data yang akan difilter

Paket data yang akan difilter terdiri dari dua macam, yaitu paket data yang menuju situs porno dan paket data yang mengandung *file* multimedia.

- Paket Data Porno

```
dest parno {
        domainlist      parno-d
        urllist         parno-u
        redirect        302:http://172.17.63.129:8080
}
```

Konfigurasi ini mendefinisikan suatu tujuan paket data bernama "parno" yang dijelaskan sebagaimana berikut :

- `domainlist` →

  Syntax ini memanggil *file* `parno-d` yang berisi daftar *domain* situs yang akan difilter oleh SquidGuard.

  Contoh : `bangbrosnetwork.com`

- `urllist` →

  Syntax ini memanggil *file* `parno-u` yang berisi daftar *url* situs yang akan difilter oleh SquidGuard.

  Contoh : `bintangmawar.net/forum`

- `redirect` →

  Syntax ini meneruskan paket data yang cocok dengan *file database* yang telah dipanggil oleh *syntax* `domainlist` dan `urllist` ke *url* `http://172.17.63.129:8080`. *Syntax* `302` adalah kode Protokol HTTP untuk melakukan *redirect* suatu *url* ke *url* yang lain.

Isi dari *file* `parno-d` adalah sebagai berikut :

```
bangbrosnetwork.com
nudecelebarchive.net
nudebabeblog.com
miriam18.com
uniformsensation.com
spermabande.de
flirt4free.com
mikesapartment.com
mindyvega.com
```

Isi dari *file* parno-u adalah sebagai berikut :

```
bintangmawar.net/forum
adult-kingdom.com/members
amateurgirls.com/restricted
```

- Paket Data Multimedia

```
dest multimedia {
        expressionlist  media
        redirect        302:http://172.17.63.129:8080
}
```

Konfigurasi ini mendefinisikan suatu tujuan paket data bernama "multimedia" yang dijelaskan sebagaimana berikut :

- `expressionlist` → *Syntax* ini memanggil *file* media yang berisi *syntax expressionlist* yang akan difilter oleh SquidGuard.
- `redirect` → *Syntax* ini meneruskan paket data yang cocok dengan *file database* yang telah dipanggil oleh *syntax* `expressionlist` ke *url* http://172.17.63.129:8080. *Syntax* `302` adalah kode HTTP untuk melakukan *redirect* suatu *url* ke *url* yang lain.

Isi dari *file* media adalah sebagai berikut :

```
(avi|mp3|mpg|3gp)$
```

f. Menentukan *Access Control List* (ACL)

```
acl {
        lasif within bebas {
                pass all
                }
        else {
                pass !in-addr !parno !multimedia all
                }
        default {
                pass            none
                redirect        http://www.google.com
        }
}
```

Konfigurasi ini adalah untuk menentukan tindakan yang dilakukan oleh SquidGuard saat melakukan pemfilteran dan langkah-langkah yang terkait

dengan proses pemfilteran. Konfigurasi-konfigurasi yang terdapat di dalam *syntax* `acl` adalah sebagai berikut :

- `lasif within bebas` →

  Syntax ini mendefinisikan bahwa suatu sumber paket data yang berasal dari "`lasif`" pada waktu "`bebas`", tidak akan difilter. *Syntax* "`pass all`" berarti bahwa semua paket data akan dilewatkan.

- `else` →

  Syntax ini mendefinisikan bahwa suatu sumber paket data yang tidak memenuhi kondisi dari *syntax* "`lasif within bebas`", akan difilter. *Syntax* "`!in-addr"`, berarti penulisan nama *host* dalam *url* harus berupa "nama *domain*" bukan berupa "nomor IP". Sedangkan *syntax* "`pass !parno !multimedia all`" berarti bahwa semua paket data selain "Paket Data Parno" dan "Paket Data Multimedia" akan dilewatkan.

- `default` →

  Syntax ini mendefinisikan bahwa suatu sumber paket data yang tidak memenuhi kondisi dari *syntax* "`lasif within bebas`" dan *syntax* "`else`", tidak akan dilewatkan dan akan di-*redirect* ke *url* `http://www.google.com`

BAB VI
PENGUJIAN SISTEM

Sistem pemfilteran layanan Internet pada *gateway* dengan menggunakan SquidGuard yang telah diimplementasikan pada Bab Lima, akan dianalisis dan diuji pada tahap ini. Pengujian dan analisis sistem perlu dilakukan untuk mengetahui apakah sistem yang telah dirancang dan diimplementasikan telah bekerja sesuai dengan tujuan pembuatan sistem. Pengujian dan analisis dilakukan secara bertahap sebagaimana urutan permasalahan yang ada di bagian rumusan masalah. Pengujian dan analisis yang dilakukan adalah pengujian dan analisis *internal network*, pengujian dan analisis sistem *gateway*, pengujian dan analisis sistem squid, pengujian dan analisis squidguard secara umum pada saat jam kerja, dan pengujian dan analisis squidguard secara umum pada saat di luar jam kerja.

Spesifikasi perangkat keras yang digunakan dalam pengujian Sistem Pemfilteran adalah sebagai berikut :

- Komputer *Gateway*
  - *Processor* : Intel Pentium 4 - 1.50 GHz
  - Motherboard : ECS P4VMM2
  - Harddisk : Maxtor 30 GB - 5400 RPM
  - *Memory* : Visipro 256 MB PC 2700
  - LAN *Card* :
    - Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+
    - D-Link System Inc RTL8139 Ethernet
  - *Graphic Card* : Eagle 64 MB nVidia RIVA TNT2
- Komputer *Client*
  - *Processor* : Intel Pentium 4 - 1.8 GHz
  - Motherboard : ECS P4VMM2
  - Harddisk : Maxtor 30 GB - 5400 RPM
  - *Memory* : Visipro 256 MB PC 2700
  - LAN *Card* : VIA PCI 10/100Mb *Fast* Ethernet *Adapter*
  - *Graphic Card* : S3 *Graphics* ProSavageDDR 32 MB *(onboard)*
  - Nomor IP : 10.100.100.7
  - Sistem Operasi : Windows XP Professional Service Pack 2

6.1. Pengujian Dan Analisis *Internal Network*

Pengujian dan analisis sistem pada tahap ini, dititikberatkan pada *internal network* yang telah diimplementasikan. Langkah-langkah pengujian adalah sebagai berikut :

a. Tujuan

Pengujian ini dilakukan untuk mengetahui apakah semua *client* dan *gateway* yang berada dalam *internal network* telah dapat saling berkomunikasi [VAS-07].

b. Spesifikasi dan Konfigurasi Komputer

Komputer *client* dan komputer *gateway* memiliki spesifikasi sebagaimana yang telah disebutkan di bagian awal bab ini. Komputer *gateway* telah dikonfigurasi sebagaimana telah dijelaskan pada Bab Empat dan Bab Lima.

c. *Software* Aplikasi

nmap-4.11-1.1

d. Prosedur Pengujian

Menjalankan perintah `nmap` pada komputer *gateway* untuk memeriksa komputer-komputer yang sedang saling terhubung dalam *internal network*.

```
[root@Konoha ~]# nmap -sP 10.100.100.0/25
```

Perintah diatas memiliki arti jalankan `nmap` dengan parameter *Ping Scan* pada *network* `10.100.100.0/25.`

e. Hasil Yang Diharapkan

Hasil yang diharapkan dari pengujian ini adalah komputer *gateway* mendapat balasan atas pesan yang dikirimkan ke semua anggota *network* `10.100.100.0/25` (semua komputer *client*).

f. Hasil Pengujian dan Analisa

Saat perintah `nmap` dijalankan, komputer *gateway* akan mengirim pesan *ping* ke seluruh anggota *network* `10.100.100.0/25.` Semua anggota *network* `10.100.100.0/25` yang pada saat perintah `nmap` dijalankan hidup, akan mengirim pesan balasan kepada komputer *gateway*. Hasil dari `nmap` adalah sebagai berikut :

```
[root@Konoha ~]# nmap -sP 10.100.100.0/25

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2007-09-20
00:38 WIT
Host 10.100.100.3 appears to be up.
Host 10.100.100.4 appears to be up.
Host 10.100.100.5 appears to be up.
Host 10.100.100.6 appears to be up.
Host 10.100.100.7 appears to be up.
Host 10.100.100.8 appears to be up.
Host 10.100.100.9 appears to be up.
Host 10.100.100.10 appears to be up.
Host 10.100.100.12 appears to be up.
Host 10.100.100.63 appears to be up.

Nmap finished: 128 IP addresses (9 hosts up) scanned in 5.638 seconds
```

Gambar 6.1 Hasil Perintah nmap

Dari hasil perintah nmap, dapat diketahui bahwa dalam *internal network* terdapat 9 *host* yang sedang hidup dengan IP 10.100.100.3 – 10.100.100.10, 10.100.100.12, dan 10.100.100.63.

g. Kesimpulan

Komputer *gateway* dan semua komputer *client* yang berada dalam *internal network* telah dapat saling berkomunikasi.

6.2. Pengujian dan Analisis Sistem *Gateway*

Pengujian dan analisis sistem pada tahap ini, dititikberatkan pada komputer *gateway*, komputer *client*, dan jaringan komputer yang menghubungkan *gateway* dan *client*.. Langkah-langkah pengujian adalah sebagai berikut :

a. Tujuan

Pengujian ini dilakukan untuk mengetahui apakah paket data yang berasal dari *internal network* dapat diteruskan oleh komputer *gateway* menuju *external network* [EPI-05].

b. Spesifikasi dan Konfigurasi Komputer

Komputer *client* dan komputer *gateway* memiliki spesifikasi sebagaimana yang telah disebutkan di bagian awal bab ini. Komputer *gateway* telah dikonfigurasi sebagaimana telah dijelaskan pada Bab Empat dan Bab Lima.

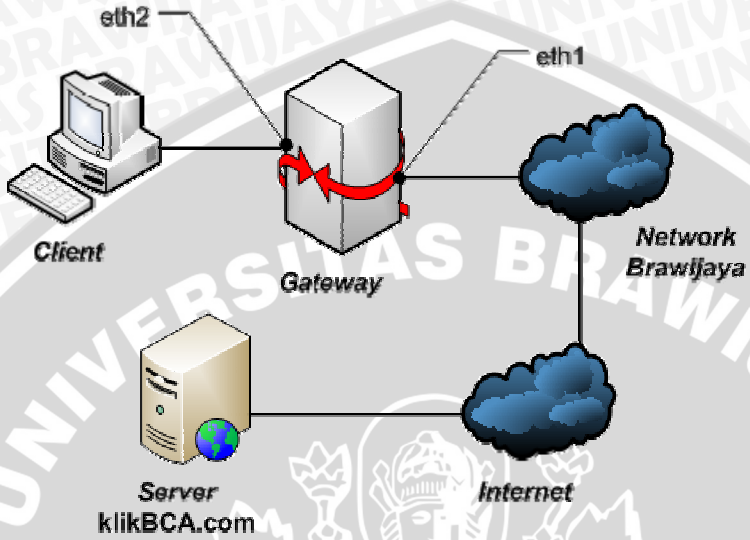
c. *Software* Aplikasi

- Mozilla Firefox 2
- tcpdump 3.9.4

- nslookup

d. Prosedur Pengujian

i. Menjalankan perintah tcpdump pada komputer *gateway* untuk melihat aliran data antara *internal network* dan *extenal network.*



Gambar 6.2 Pengujian *Gateway*

Dua perintah tcpdump dijalankan secara paralel untuk melihat paket data yang bersumber dan bertujuan ke IP 10.100.100.7. Perintah tcpdump yang berfungsi untuk melihat paket data yang bersumber dari IP 10.100.100.7 dan bersumber dari *port* 80 adalah sebagai berikut :

```
[root@Konoha ~]# tcpdump -ni eth2 ip src
10.100.100.7 and dst port 80
```

Perintah diatas memilki arti jalankan perintah tcpdump tanpa melakukan *resolve name server* pada *interface eth2* dan khususkan pada paket data yang bersumber dari IP 10.100.100.7 dan bertujuan ke *port* 80.

Perintah tcpdump yang berfungsi untuk melihat paket data yang bertujuan ke IP 10.100.100.7 adalah sebagai berikut :

```
[root@Konoha ~]# tcpdump -ni eth2 ip dst
10.100.100.7
```

Perintah diatas memilki arti jalankan perintah tcpdump *resolve name server* pada *interface eth2* dan khususkan pada paket data yang bersumber dari IP 10.100.100.7

ii. Menjalankan Firefox yang ada di komputer *client*.

→ Start | All Program | Mozilla Firefox | Mozilla Firefox

iii. Membuka situs www.klikbca.com dengan Firefox 2

e. Hasil Yang Diharapkan

Hasil yang diharapkan dari pengujian ini adalah Firefox 2 di komputer *client* dapat menampilkan situs www.klikbca.com.

f. Hasil Pengujian dan Analisa

Saat komputer *client* menjalankan Firefox 2 dan membuka situs www.klikbca.com, tcpdump yg sedang berjalan di komputer *gateway* akan merekam paket data yang mengalir antara komputer *client* dengan server www.klikbca.com. Hasil dari tcpdump adalah sebagai berikut :

```
[root@Konoha ~]# tcpdump -ni eth2 ip src 10.100.100.7 and dst port 80

tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes

15:45:19.158893 IP 10.100.100.7.de-server > 202.6.211.8.http: S
        3864355344:3864355344(0) win 65535 <mss 1460,nop,nop,sackOK>
15:45:19.161921 IP 10.100.100.7.de-server > 202.6.211.8.http: . ack
        2580751890 win 65535
15:45:19.187061 IP 10.100.100.7.de-server > 202.6.211.8.http: P
        0:496(496) ack 1 win 65535
.
.
.

15:45:34.360425 IP 10.100.100.7.opennl-voice > 202.6.211.8.http: . ack
        655 win 64881
15:45:34.575740 IP 10.100.100.7.opennl-voice > 202.6.211.8.http: . ack
        655 win 64881

53 packets captured
106 packets received by filter
0 packets dropped by kernel
```

Gambar 6.3 Hasil tcpdump Paket Data yang Bersumber Dari IP 10.100.100.7 dan Bertujuan ke *Port* 80 pada Pengujian Sistem *Gateway*

Dari hasil tcpdump, dapat diketahui bahwa ada paket data dari IP 10.100.100.7 menuju IP 202.6.211.8 dengan *port* http (80). Keterangan lebih detail dari hasil perintah tcpdump diatas adalah sebagai berikut :

- 53 packets captured berarti saat perintah tcpdump dihentikan, terdapat 53 paket data yang berhasil dicatat oleh tcpdump.
- 106 packets received by filter berarti saat perintah tcpdump dihentikan, terdapat 106 paket data yang sesuai dengan parameter yang diberikan pada tcpdump.
- 0 packets dropped by kernel berarti tidak ada paket data yang terbuang akibat buffer penuh.

```
[root@Konoha ~]# tcpdump -ni eth2 ip dst 10.100.100.7

tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes

23:30:05.473683 IP 202.6.211.8.http > 10.100.100.7.36200: S
        585278217:585278217(0) ack 3187407279 win 5792 <mss
        1460,sackOK,timestamp 130861582 9329149,nop,wscale 2>
23:30:05.479147 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 400 win
        1716 <nop,nop,timestamp 130861586 9329150>
23:30:06.402907 IP 202.6.211.8.http > 10.100.100.7.36200: P
        1:1390(1389) ack 400 win 1716 <nop,nop,timestamp 130862507
        9329150>
.
.
.
23:30:40.076924 IP 202.6.211.8.http > 10.100.100.7.36204: P
        5334:6702(1368) ack 1353 win 2252 <nop,nop,timestamp 130896198
        9337789>
23:30:40.622651 IP 202.6.211.8.http > 10.100.100.7.36204: P
        6702:6758(56) ack 1353 win 2252 <nop,nop,timestamp 130896205
        9337789>

174 packets captured
348 packets received by filter
0 packets dropped by kernel
```

Gambar 6.4 Hasil tcpdump Paket Data yang Bertujuan ke IP 10.100.100.7 pada Pengujian Sistem *Gateway*

Dari hasil tcpdump, dapat diketahui bahwa ada paket data dari IP 202.6.211.8 dengan *port* http (80) menuju IP 10.100.100.7 dengan *port* http (80). Keterangan lebih detail dari hasil perintah tcpdump diatas adalah sebagai berikut :

- `174 packets captured` berarti saat perintah `tcpdump` dihentikan, terdapat 174 paket data yang berhasil dicatat oleh `tcpdump`.
- `348 packets received by filter` berarti saat perintah `tcpdump` dihentikan, terdapat 348 paket data yang sesuai dengan parameter yang diberikan pada `tcpdump`.
- `0 packets dropped by kernel` berarti tidak ada paket data yang terbuang akibat *buffer* penuh.

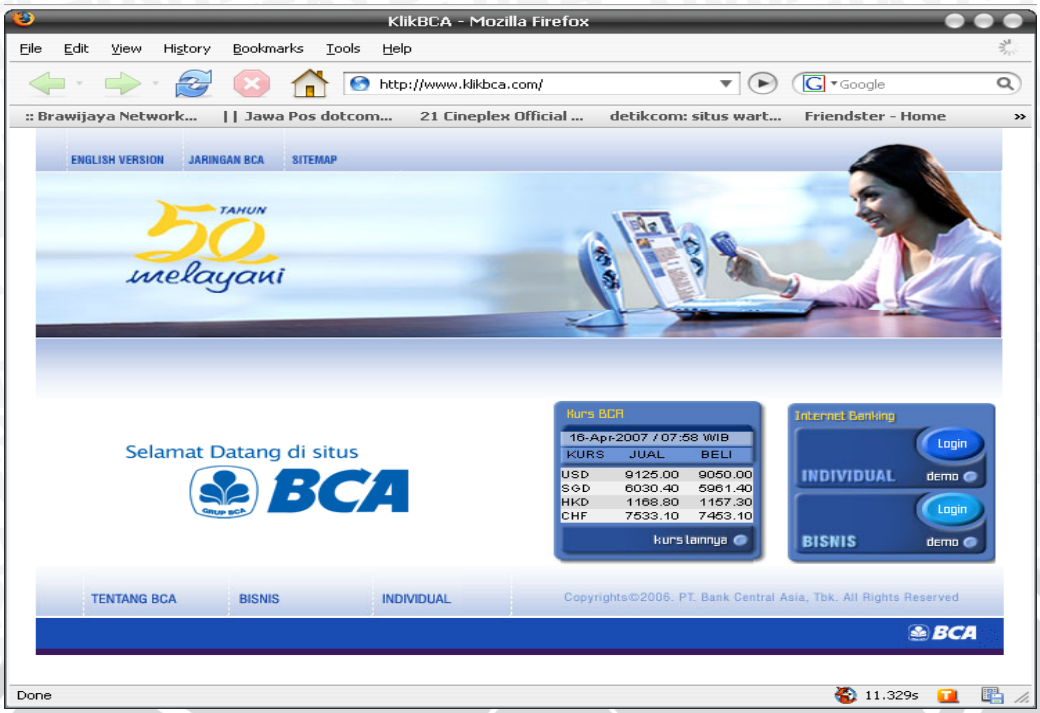
```
[root@Konoha ~]# nslookup www.klikbca.com
Server:         202.162.208.99
Address:        202.162.208.99#53

Non-authoritative answer:
Name:   www.klikbca.com
Address: 202.6.211.8
```

Dengan perintah `nslookup`, dapat diketahui bahwa `www.klikbca.com` memiliki nomor IP 202.6.211.8. Keterangan lebih detail dari hasil perintah `nslookup` diatas adalah sebagai berikut :

- `Server : 202.162.208.99` berarti *Domain Name Server* yang digunakan adalah `202.162.208.99`
- `Address: 202.162.208.99#53` berarti *Domain Name Server* yang digunakan memilki alamat `202.162.208.99` dengan *port* 53.
- `Non-authoritative answer` berarti yang memberikan jawaban bukan *Domain Name Server* resmi yang ditunjuk oleh *server* yang bersangkutan.
- `Name : www.klikbca.com` berarti nama dari *server* yang akan dicari nomor IP-nya
- `Address: 202.6.211.8` berarti nomor IP dari *server* yang dicari adalah `202.6.211.8`

Dalam Gambar 6.5 diperlihatkan bahwa alamat `www.klikbca.com` telah berhasil dibuka dengan menggunakan Firefox 2.



Gambar 6.5 Firefox 2 menampilkan situs www.klikbca.com

g. Kesimpulan

Paket data yang berasal dari *internal network*, yaitu paket data yang berasal dari IP 10.100.100.3 dapat diteruskan dengan baik oleh komputer *gateway* menuju ke *external network* dengan alamat www.klikbca.com yang memilki IP 202.6.211.8.

6.3. Pengujian dan Analisis Squid

Pengujian dan analisis sistem pada tahap ini, dititikberatkan pada Squid terpasang di dalam komputer *gateway*. Langkah-langkah pengujian adalah sebagai berikut :

a. Tujuan

Pengujian ini dilakukan untuk mengetahui apakah Squid yang telah terpasang dan telah dikonfigurasi, dapat berjalan dengan baik dan tidak melakukan *cache web content* [TEX-07].

b. Spesifikasi dan Konfigurasi Komputer

Komputer client dan komputer gateway memiliki spesifikasi sebagaimana yang telah disebutkan di bagian awal bab ini. Komputer gateway telah dikonfigurasi sebagaimana telah dijelaskan pada Bab Empat dan Bab Lima.

c. Software Aplikasi

- Squid 2.6.STABLE4-1.fc6
- Mozilla Firefox 2
- tcpdump 3.9.4
- nslookup
- iptables 1.3.5-1.2.1
- iptables-save v1.3.5
- tail (GNU coreutils) 5.97

d. Prosedur Pengujian

i. Mengaktifkan iptables di komputer *gateway*

```
[root@Konoha ~]# /etc/init.d/iptables start
Flushing firewall rules:                   [  OK  ]
Setting chains to policy ACCEPT: filter nat [  OK  ]
Unloading iptables modules:                [  OK  ]
Applying iptables firewall rules:          [  OK  ]
Loading additional iptables modules:
ip_conntrack_netbios_n_ftp                 [  OK  ]
```

ii. Mengaktifkan Squid di komputer *gateway*

```
[root@Konoha ~]# /etc/init.d/squid start
Starting squid: .                          [  OK  ]
```

iii. Membuka situs www.bni.co.id dengan Firefox 2 di komputer *client.*

e. Hasil Yang Diharapkan

Hasil yang diharapkan dari pengujian ini adalah Firefox 2 di komputer *client* dapat menampilkan situs www.bni.co.id dan di dalam *file* /var/log/squid/store.log, terdapat keterangan "RELEASE".

f. Hasil Pengujian dan Analisa

Hasil pengujian dan analisa dilakukan pada tiap tahap prosedur pengujian. Hasil pengujian dan analisa dijelaskan sebagaimana berikut :

i. Status iptables

Untuk dapat mengetahui status iptables, maka perintah iptables-save dapat digunakan.

```
[root@Konoha ~]# iptables-save
# Generated by iptables-save v1.3.5 on Wed Apr 18
17:29:54 2007
*filter
:INPUT DROP [639:110479]
:FORWARD DROP [54:2592]
:OUTPUT ACCEPT [60:7676]
-A FORWARD -m state --state RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -s 10.100.100.0/255.255.255.128 -i eth2 -
o eth1 -p udp -m udp –dport 53 -m state --state NEW
-j ACCEPT
-A FORWARD -s 10.100.100.0/255.255.255.128 -i eth2 -
o eth1 -p tcp -m multiport --dports
20,21,443,1863,3128,5050,5061,8080,33444 -m state --
state NEW -j ACCEPT
COMMIT
# Completed on Wed Apr 18 17:29:54 2007
# Generated by iptables-save v1.3.5 on Wed Apr 18
17:29:54 2007
*nat
:PREROUTING ACCEPT [774:125308]
:POSTROUTING ACCEPT [1:92]
:OUTPUT ACCEPT [1:92]
-A PREROUTING -s 10.100.100.0/255.255.255.128 -i
eth2 -p tcp -m tcp --dport 80 -j REDIRECT --to
-ports 3128
-A PREROUTING -s 10.100.100.0/255.255.255.128 -i
eth2 -p tcp -m tcp --dport 33444 -j REDIRECT --to
-ports 3128
COMMIT
# Completed on Wed Apr 18 17:29:54 2007
```

Gambar 6.6 Hasil Perintah iptables-save

Dari hasil perintah iptables-save, dapat diketahui bahwa semua konfigurasi seperti perintah-perintah FORWARD dan PREROUTING dapat ditampilkan maka dapat disimpulkan bahwa iptables sedang dijalankan oleh sistem.

ii. Status Squid

Untuk mengetahui apakah Squid telah berjalan, dapat dilihat dengan perintah sebagai berikut :

```
[root@Konoha ~]# tail -f cache.log
  2007/04/20 13:16:16| Starting Squid Cache version 2.6.STABLE4 for
                       i686-redhat-linux-gnu...
  2007/04/20 13:16:16| Process ID 4681
  2007/04/20 13:16:16| With 1024 file descriptors available
  2007/04/20 13:16:16| Using epoll for the IO loop
  2007/04/20 13:16:16| DNS Socket created at 0.0.0.0, port 32775, FD 5
  2007/04/20 13:16:16| Adding domain brawijaya.ac.id from
                       /etc/resolv.conf
  2007/04/20 13:16:16| Adding nameserver 202.162.208.99 from
                       /etc/resolv.conf
  2007/04/20 13:16:16| Adding nameserver 202.162.208.100 from
                       /etc/resolv.conf
  2007/04/20 13:16:16| helperOpenServers: Starting 25 'squidGuard'
                       processes
  2007/04/20 13:16:16| User-Agent logging is disabled.
  2007/04/20 13:16:16| Referer logging is disabled.
  2007/04/20 13:16:16| Unlinkd pipe opened on FD 35
  2007/04/20 13:16:16| Swap maxSize 51200 KB, estimated 3938 objects
  2007/04/20 13:16:16| Target number of buckets: 196
  2007/04/20 13:16:16| Using 8192 Store buckets
  2007/04/20 13:16:16| Max Mem  size: 32768 KB
  2007/04/20 13:16:16| Max Swap size: 51200 KB
  2007/04/20 13:16:16| Local cache digest enabled; rebuild/rewrite every
                       3600/3600 sec
  2007/04/20 13:16:16| Rebuilding storage in /var/spool/squid (CLEAN)
  2007/04/20 13:16:16| Using Least Load store dir selection
  2007/04/20 13:16:16| Set Current Directory to /var/spool/squid
  2007/04/20 13:16:16| Loaded Icons.
  2007/04/20 13:16:17| Accepting transparently proxied HTTP connections
                       at 0.0.0.0, port 3128, FD 37.
  2007/04/20 13:16:17| Accepting ICP messages at 0.0.0.0, port 3130, FD
                       38.
  2007/04/20 13:16:17| WCCP Disabled.
  2007/04/20 13:16:17| Ready to serve requests.
  2007/04/20 13:16:17| Done reading /var/spool/squid swaplog (0 entries)
  2007/04/20 13:16:17| Finished rebuilding storage from disk.
  2007/04/20 13:16:17|         0 Entries scanned
  2007/04/20 13:16:17|         0 Invalid entries.
  2007/04/20 13:16:17|         0 With invalid flags.
  2007/04/20 13:16:17|         0 Objects loaded.
  2007/04/20 13:16:17|         0 Objects expired.
  2007/04/20 13:16:17|         0 Objects cancelled.
  2007/04/20 13:16:17|         0 Duplicate URLs purged.
  2007/04/20 13:16:17|         0 Swapfile clashes avoided.
  2007/04/20 13:16:17|   Took 0.3 seconds (   0.0 objects/sec).
  2007/04/20 13:16:17| Beginning Validation Procedure
  2007/04/20 13:16:17|   Completed Validation Procedure
  2007/04/20 13:16:17|   Validated 0 Entries
  2007/04/20 13:16:17|   store_swap_size = 0k
  2007/04/20 13:16:17| storeLateRelease: released 0 objects
```

Gambar 6.7 Isi Dari *File* /var/log/squid/cache.log

Dari isi *file* /var/log/squid/cache.log, tampak bahwa "`Ready to serve requests`" yang berarti Squid telah siap melayani *request* dari komputer *client.*

iii. Analisa kerja Squid saat menerima paket data dari komputer *client*

Saat komputer *client* menjalankan Firefox 2 dan membuka situs www.bni.co.id, Firefox 2 akan mengirimkan paket data *request* ke *url*

www.bni.co.id dengan port tujuan 80. Saat Paket data ini sampai di komputer *gateway*, iptables akan membelokkan (me-*redirect*) port tujuan paket data dari 80 menjadi 3128.

Pengamatan paket data dengan menggunakan tcpdump akan memberikan hasil sebagaimana berikut :

```
[root@Konoha squid]# tcpdump -nni eth2 ip src 10.100.100.7 and dst
port 80

tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96
bytes

08:08:28.300994 IP 10.100.100.7.44838 > 219.83.38.10.80: S
        829824112:829824112(0) win 5840 <mss 1460,sackOK,timestamp
        10804818 0,nop,wscale 5>
08:08:28.301088 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack
        1920127707 win 183 <nop,nop,timestamp 10804818 15461601>
08:08:28.301724 IP 10.100.100.7.44838 > 219.83.38.10.80: P
        0:397(397) ack 1 win 183 <nop,nop,timestamp 10804818
        15461601>
.
.
.
08:08:30.493105 IP 10.100.100.7.44842 > 219.83.38.10.80: . ack 1602
        win 364 <nop,nop,timestamp 10805366 15462149>
08:08:30.930840 IP 10.100.100.7.44842 > 219.83.38.10.80: . ack 2079
        win 454 <nop,nop,timestamp 10805476 15462258>

37 packets captured
74 packets received by filter
0 packets dropped by kernel
```

Gambar 6.8 Hasil tcpdump Paket Data yang Bersumber Dari IP 10.100.100.7 dan Bertujuan ke *Port* 80 pada Pengujian Squid

```
[root@Konoha squid]# tcpdump -nni eth2 ip dst 10.100.100.7

tcpdump: verbose output suppressed, use -v or -vv for full protocol
        decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96
        bytes
01:08:28.301386 IP 219.83.38.10.80 > 10.100.100.7.44838: S
        1920127706:1920127706(0) ack 829824113 win 5792 <mss
        1460,sackOK,timestamp 15461601 10804818,nop,wscale 5>
01:08:28.301824 IP 219.83.38.10.80 > 10.100.100.7.44838: . ack 398
        win 215 <nop,nop,timestamp 15461601 10804818>
01:08:28.314455 IP 219.83.38.10.80 > 10.100.100.7.44838: .
        1:1449(1448) ack 398 win 215 <nop,nop,timestamp 15461604
        10804818>
.
.
.

01:08:30.492612 IP 219.83.38.10.80 > 10.100.100.7.44842: P
        1449:1602(153) ack 381 win 215 <nop,nop,timestamp 15462149
        10804996>
```

```
01:08:30.930541 IP 219.83.38.10.80 > 10.100.100.7.44842: P
        1602:2079(477) ack 381 win 215 <nop,nop,timestamp 15462258
        10805366>

32 packets captured
64 packets received by filter
0 packets dropped by kernel
```

Gambar 6.9 Hasil tcpdump Paket Data yang Bertujuan ke IP 10.100.100.7 pada Pengujian Squid

Dari hasil perintah tcpdump dapat diketahui bahwa terdapat paket data dari IP 10.100.100.7 yang menghubungi IP 219.83.38.10 dengan *port* tujuan 80. Dengan menggunakan perintah nslookup, dapat diketahui bahwa IP 219.83.38.10 merupakan IP dari *url* www.bni.co.id.

```
[root@Konoha ~]# nslookup www.bni.co.id
Server:         202.162.208.99
Address:        202.162.208.99#53

Non-authoritative answer:
Name:   www.bni.co.id
Address: 219.83.38.10
```

Untuk mengetahui apakah paket data dari IP 10.100.100.7 menghubungi IP 219.83.38 melalui Squid, dapat dilakukan dengan cara melihat isi *file* /var/log/squid/access.log.

```
[root@Konoha squid]# tail -f access.log

1177832039.466   1087 10.100.100.7 TCP_MISS/200 2434 GET
      http://www.bni.co.id/ - DIRECT/219.83.38.10 text/html
1177832040.580   1113 10.100.100.7 TCP_MISS/200 1801 GET
      http://www.bni.co.id/splash.css - DIRECT/219.83.38.10
      text/css
1177832040.595     14 10.100.100.7 TCP_MISS/302 154 GET
      http://www.bni.co.id/favicon.ico - NONE/- -
1177832041.649    977 10.100.100.7 TCP_MISS/200 770 GET
      http://www.bni.co.id/imgsplash/shade.jpg -
      DIRECT/219.83.38.10 image/jpeg
1177832042.674   2007 10.100.100.7 TCP_MISS/200 4033 GET
      http://www.bni.co.id/imgsplash/bni_logo.gif -
      DIRECT/219.83.38.10 image/gif
1177832042.675   2000 10.100.100.7 TCP_MISS/200 3653 GET
      http://www.bni.co.id/imgsplash/reflection.gif -
      DIRECT/219.83.38.10 image/gif
1177832044.830   4157 10.100.100.7 TCP_MISS/200 2066 GET
      http://www.bni.co.id/imgsplash/link_bg.gif -
      DIRECT/219.83.38.10 image/gif
```

Gambar 6.10 Isi Dari *File* /var/log/squid/access.log pada Pengujian Squid

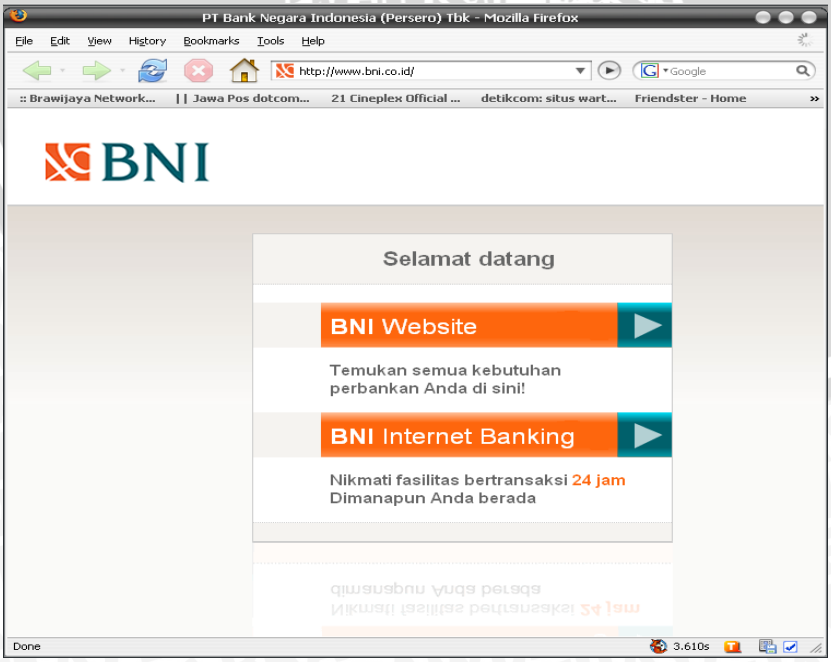
Dari isi *file* access.log, dapat disimpulkan bahwa paket data dari IP 10.100.100.7 pergi menuju IP 219.83.38.10 dengan perantara Squid. Tampak dalam Gambar 6.12, situs www.bni.co.id berhasil dibuka oleh komputer *client* dengan Mozilla Firefox 2.

```
[root@Konoha squid]# tail -f store.log

1177832039.466 RELEASE -1 FFFFFFFF EC97604A7CC142451E2E4645B7B24404
      200 1175653909 1175492248        -1 text/html 1933/1933 GET
      http://www.bni.co.id/
1177832040.580 RELEASE -1 FFFFFFFF C81C5BF1C7DCF34D144894D0AAA99154
      200 1175653915 1175492732        -1 text/css 1353/1353 GET
      http://www.bni.co.id/splash.css
1177832040.595 RELEASE -1 FFFFFFFF 3E6438CBD16F5482F38739B2BC1F48A2
      302 1177832040        -1        -1 unknown 0/0 GET
      http://www.bni.co.id/favicon.ico
1177832041.649 RELEASE -1 FFFFFFFF CA6572C2CB8FB1397BD89A1288E0142D
      200 1175653924 1175492248        -1 image/jpeg 322/322 GET
      http://www.bni.co.id/imgsplash/shade.jpg
1177832042.674 RELEASE -1 FFFFFFFF 9BA4D44D44FCC8D8F683F054ADA9B80E
      200 1175653915 1175492247        -1 image/gif 3584/3584 GET
      http://www.bni.co.id/imgsplash/bni_logo.gif
1177832042.675 RELEASE -1 FFFFFFFF 5F6721BF58134844B88FD8DE0266AA27
      200 1175653915 1175492248        -1 image/gif 3206/3206 GET
      http://www.bni.co.id/imgsplash/reflection.gif
1177832044.830 RELEASE -1 FFFFFFFF 9021162358CAAFEF65EF016ED3EEFA47
      200 1175653924 1175492247        -1 image/gif 1617/1617 GET
      http://www.bni.co.id/imgsplash/link_bg.gif
```

Gambar 6.11 Isi Dari *File* /var/log/squid/store.log

Di dalam *file* /var/log/squid/store.log, terdapat keterangan "RELEASE" yang menunjukkan bahawa Squid tidak melakukan *cache web content* terhadap situs www.bni.co.id.



Gambar 6.12 Firefox 2 menampilkan situs www.bni.co.id

g. Kesimpulan

Komputer *client* dapat mengakses *url* www.bni.co.id dengan sukses melalui perantara Squid dengan menggunakan Mozilla Firefox 2. Port tujuan dari paket data komputer *client* berhasil diarahkan oleh iptables dari 80 menjadi 3128. Squid berhasil meneruskan paket data dari komputer *client* dan meneruskan paket data balasan kembali ke komputer *client* tanpa melakukan *cache web content.*

6.4. Pengujian dan Analisis SquidGuard Secara Umum pada saat Jam Kerja

Pengujian dan analisis konfigurasi SquidGuard secara umum pada saat jam kerja, maksudnya adalah SquidGuard diuji langsung dengan komputer *client*. Dengan *browser* Mozilla Firefox 2, komputer *client* mengakses suatu situs dengan melewati SquidGuard. Langkah-langkah pengujian adalah sebagai berikut :

a. Tujuan

Pengujian ini dilakukan untuk mengetahui apakah SquidGuard telah berjalan sesuai dengan konfigurasi yang telah diberikan, saat digunakan dalam kondisi dan waktu secara umum pada saat jam kerja. Kondisi dan waktu secara umum berarti SquidGuard memproses paket-paket data dari komputer *client* yang mengakses suatu situs dengan menggunakan *browser* [KOR-07].

b. Spesifikasi dan Konfigurasi Komputer

Komputer *client* dan komputer *gateway* memiliki spesifikasi sebagaimana yang telah disebutkan di bagian awal bab ini. Komputer *gateway* telah dikonfigurasi sebagaimana telah dijelaskan pada Bab Empat dan Bab Lima.

c. Software Aplikasi

- tcpdump 3.9.4
- Squid 2.6.STABLE4-1.fc6
- SquidGuard 1.2.0-14.fc6
- Mozilla Firefox 2
- tail (GNU coreutils) 5.97

d. Prosedur Pengujian

i. Mengaktifkan iptables, Squid dan SquidGuard di komputer *gateway*

Prosedur untuk mengaktifkan iptables dan Squid telah dijelaskan pada bagian "**6.3 Pengujian dan Analisis Squid**". SquidGuard merupakan *plug-in* dari

Squid, sehingga saat Squid di aktifkan maka SquidGuard secara otomatis ikut aktif.

ii. Mengakses situs yang tidak diblokir pada saat jam kerja

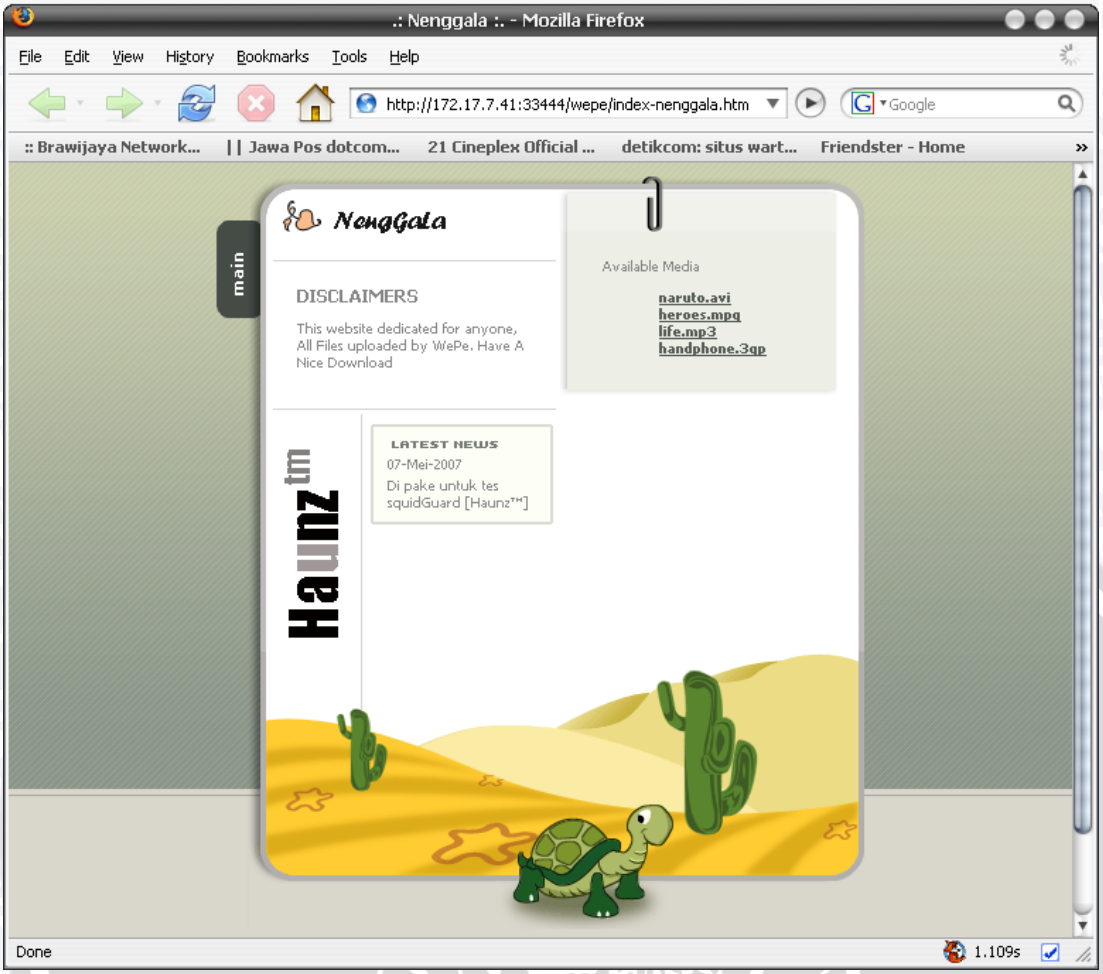
Situs tak terblokir yang akan diakses pada saat jam kerja adalah www.national-anime.com. Situs tersebut dibuka dengan Firefox 2 di komputer *client.*

iii. Mengakses *web content* yang diblokir pada saat jam kerja

Dalam prosedur ini, komputer *client* akan membuka beberapa *web content* yang diblokir pada saat jam kerja. *Web content* tersebut adalah sebagai berikut :

- bangbrosnetwork.com
- bintangmawar.net/forum
- *File* ber-*extention* mp3, avi, mpg, 3gp yang akan di-*download* dari http://172.17.7.41:33444/wepe/index-nenggala.htm.

Tampilan dari alamat http://172.17.7.41:33444/wepe/index-nenggala.htm diperlihatkan dalam Gambar 6.13.



Gambar 6.13 Firefox 2 Menampilkan Alamat http://172.17.7.41:33444/wepe/index-nenggala.htm

e. Hasil Yang Diharapkan

Hasil yang diharapkan dari pengujian ini adalah pada saat jam kerja, yaitu Senin sampai Jum'at pukul 07.30 hingga pukul 15.30, Firefox 2 di komputer *client* dapat menampilkan situs www.national-anime.com, http://172.17.63.129:8080, dan www.brawijaya.ac.id

f. Hasil Pengujian dan Analisa

Hasil pengujian dan analisa dilakukan pada tiap tahap prosedur pengujian. Hasil pengujian dan analisa dijelaskan sebagaimana berikut :

i. Status SquidGuard

Untuk mengetahui apakah SquidGuard telah aktif dan siap memproses paket data, dapat dilakukan dengan cara melihat isi dari *file* /var/log/squid/squidGuard.log.

```
[root@Konoha squid]# tail -f squidGuard.log

2007-05-01 14:36:57 [17244] New setting: dbhome: /var/squidGuard/cream
2007-05-01 14:36:57 [17244] New setting: logdir: /var/log/squid
2007-05-01 14:36:57 [17244] init domainlist
                            /var/squidGuard/cream/parno-d
2007-05-01 14:36:57 [17244] loading dbfile
                            /var/squidGuard/cream/parno-d.db
2007-05-01 14:36:57 [17253] init urllist /var/squidGuard/cream/parno-u
2007-05-01 14:36:57 [17253] loading dbfile
                            /var/squidGuard/cream/parno-u.db
2007-05-01 14:36:57 [17254] init urllist /var/squidGuard/cream/parno-u
2007-05-01 14:36:57 [17254] loading dbfile
                            /var/squidGuard/cream/parno-u.db
2007-05-01 14:36:57 [17254] init expressionlist
                            /var/squidGuard/cream/media
2007-05-01 14:36:57 [17232] squidGuard 1.2.0 started (1178005017.821)
2007-05-01 14:36:57 [17232] recalculating alarm in 3243 seconds
2007-05-01 14:36:57 [17232] squidGuard ready for requests
                            (1178005017.918)
2007-05-01 14:36:57 [17238] init urllist /var/squidGuard/cream/parno-u
2007-05-01 14:36:57 [17238] loading dbfile
                            /var/squidGuard/cream/parno-u.db
2007-05-01 14:36:57 [17238] init expressionlist
                            /var/squidGuard/cream/media
2007-05-01 14:36:57 [17238] squidGuard 1.2.0 started (1178005017.857)
2007-05-01 14:36:57 [17238] recalculating alarm in 3243 seconds
2007-05-01 14:36:57 [17238] squidGuard ready for requests
                            (1178005017.920)
```

Gambar 6.14 Isi Dari *File* /var/log/squid/squidGuard.log

Di dalam file squidGuard.log, tampak bahwa "`squidGuard ready for requests`" yang berarti SquidGuard telah aktif dan siap memproses paket data.

ii. Akses situs yang tidak diblokir pada saat jam kerja

Dalam Gambar 6.16 tampak bahwa www.national-anime.com telah berhasil dibuka pada saat SquidGuard sedang aktif. Paket data komputer client yang mengakses www.national-anime.com tampak dalam file /var/log/squid/access.log. Isi dari *file* access.log saat membuka www.national-anime.com adalah sebagai berikut :

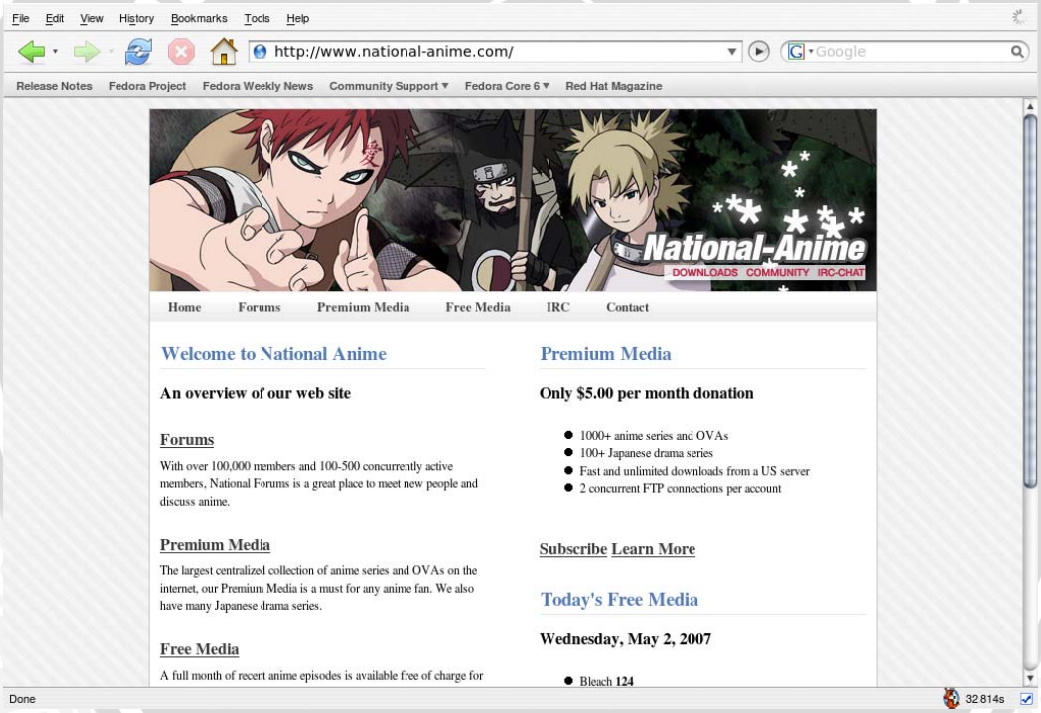
```
[root@Konoha squid]# tail -f access.log

1178181938.922   7050 10.100.100.7 TCP_MISS/200 4739 GET
                 http://www.national-anime.com/ -
                 DIRECT/208.122.8.190 text/html
1178181940.125   4446 10.100.100.7 TCP_MISS/200 2089 GET
                 http://www.national-anime.com/all.css -
                 DIRECT/208.122.8.190 text/css
1178181943.970   3699 10.100.100.7 TCP_MISS/200 582 GET
                 http://www.national-anime.com/images/menu_bg.gif -
                 DIRECT/208.122.8.190 image/gif
```

```
1178181944.027   3902 10.100.100.7 TCP_MISS/200 580 GET
                 http://www.national-anime.com/images/body-
                 background.gif - DIRECT/208.122.8.190 image/gif
1178181963.442  23315 10.100.100.7 TCP_MISS/200 51807 GET
                 http://www.national-anime.com/images/banner.jpg -
                 DIRECT/208.122.8.190 image/jpeg
1178181976.757  13292 10.100.100.7 TCP_MISS/302 625 GET
                 http://www.national-anime.com/favicon.ico -
                 DIRECT/208.122.8.190 text/html
1178182999.091  12724 10.100.100.7 TCP_MISS/200 5956 GET
                 http://sb.google.com/safebrowsing/update? -
                 DIRECT/72.14.253.93 text/plain
```

Gambar 6.15 Isi Dari *File* /var/log/squid/access.log pada Pengujian SquidGuard secara Umum

Dari isi *file* access.log, dapat disimpulkan bahwa paket data dari IP 10.100.100.7 pergi menuju IP 208.122.8.190 dengan perantara Squid. Hasil pengaksesan situs tersebut ditunjukan dalam Gambar 6.16.



Gambar 6.16 menampilkan situs www.national-anime.com

iii. Akses *web content* yang diblokir pada saat jam kerja

- Akses alamat bangbrosnetwork.com

  Akses ke alamat bangbrosnetwork.com yang dilakukan oleh komputer *client* akan tampak dalam *file* /var/log/squid/access.log dan /var/log/squid/parno.log. Isi dari *file* access.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1178611417.285      1 10.100.100.7 TCP_MISS/302 154 GET
                  http://www.bangbrosnetwork.com/ - NONE/- -
```

Gambar 6.17 Isi Dari *File* /var/log/squid/access.log saat Mengakses www.bangbrosnetwork.com

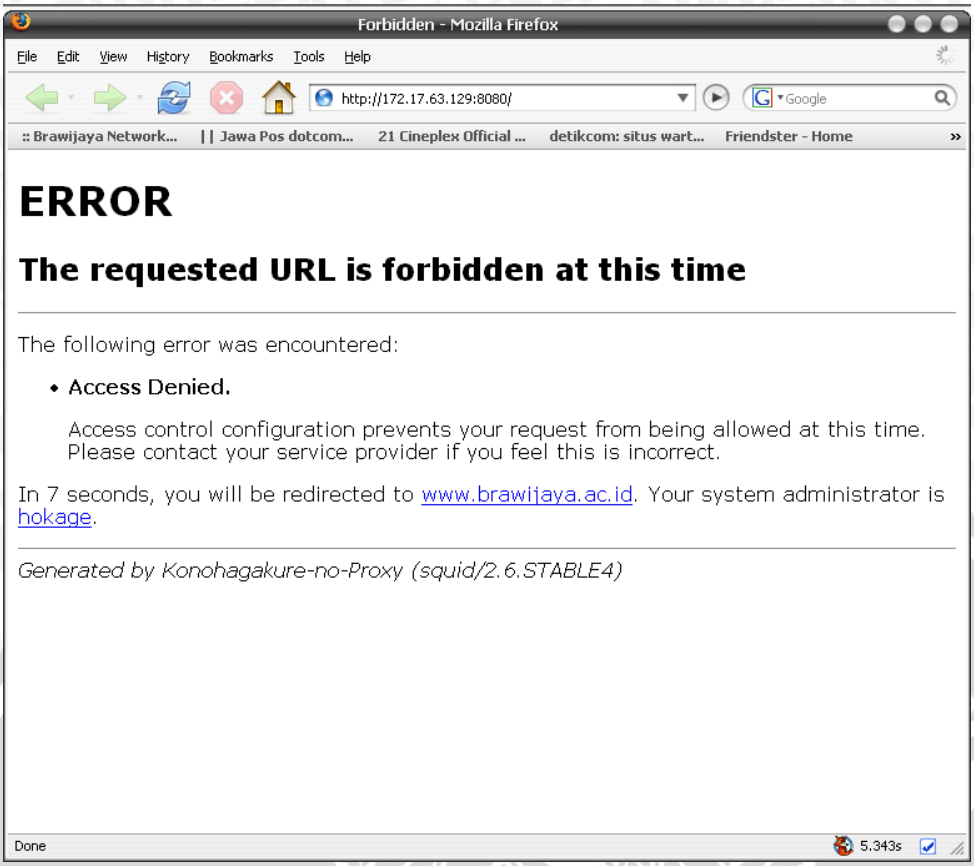
Isi dari *file* parno.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f parno.log
2007-05-08 15:06:48 [31487] Request(lasif/parno/-)
          http://www.bangbrosnetwork.com/ 10.100.100.7/- - GET
          REDIRECT
```

Gambar 6.18 Isi Dari *File* /var/log/squid/parno.log saat Mengakses www.bangbrosnetwork.com

Berdasarkan isi dari *file* access.log yang menunjukkan adanya keterangan "NONE" dan parno.log yang menunjukkan adanya keterangan "GET REDIRECT", dapat diketahui bahwa paket data dari komputer *client* saat melewati Squid diarahkan ke alamat lain. Paket data yang akan menuju bangbrosnetwork.com diarahkan ke alamat http://172.17.63.129:8080.

Pada alamat http://172.17.63.129:8080 , terdapat informasi bahwa alamat yang diakses oleh pengguna tidak dapat diakses saat ini dan dalam 7 detik, halaman tersebut akan ter-*redirect* ke alamat www.brawijaya.ac.id. Alamat http://172.17.63.129:8080 ditampilkan dalam Gambar 6.19 dan alamat www.brawijaya.ac.id ditampilkan dalam Gambar 6.20.



Gambar 6.19 Firefox 2 menampilkan situs http://172.17.63.129:8080

Gambar 6.20 Firefox 2 menampilkan situs www.brawijaya.ac.id

- Akses alamat bintangmawar.net/forum

  Saat komputer *client* mengakses alamat www.bintangmawar.net, Firefox 2 berhasil menampilkan isi dari alamat tersebut karena memang SquidGuard tidak melakukan pemblokiran terhadap *domain* www.bintangmawar.net. Isi dari *file* access.log saat membuka www.bintangmawar.net adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1178617423.368      8 10.100.100.7 TCP_MISS/302 154 GET
                http://www.bintangmawar.net/favicon.ico - NONE/- -
1178617426.003   5710 10.100.100.7 TCP_MISS/200 8529 GET
                http://www.bintangmawar.net/ -
                DIRECT/63.243.152.34 text/html
1178617427.614   4059 10.100.100.7 TCP_MISS/200 6405 GET
                http://www.bintangmawar.net/i/warning.gif -
                DIRECT/63.243.152.34 image/gif
1178617428.416   4025 10.100.100.7 TCP_MISS/200 477 GET
                http://www.bintangmawar.net/i/shim.gif -
                DIRECT/63.243.152.34 image/gif
1178617431.795   5775 10.100.100.7 TCP_MISS/200 1282 GET
                http://www.bintangmawar.net/i/nav_enter.gif -
                DIRECT/63.243.152.34 image/gif
1178617432.316   6297 10.100.100.7 TCP_MISS/200 4816 GET
                http://www.bintangmawar.net/i/logo.gif -
                DIRECT/63.243.152.34 image/gif
1178617433.301   6491 10.100.100.7 TCP_MISS/200 519 GET
                http://ads.bintangmawar.net/adx.js -
                DIRECT/63.243.152.34 application/x-javascript
1178617440.353   7051 10.100.100.7 TCP_MISS/200 1835 GET
                http://ads.bintangmawar.net/adjs.php? -
                DIRECT/63.243.152.34 application/x-javascript
1178617441.034    629 10.100.100.7 TCP_MISS/200 3713 GET
                http://www.statcounter.com/counter/frames.js -
                DIRECT/64.246.52.3 application/x-javascript
1178617444.803   4449 10.100.100.7 TCP_MISS/200 470 GET
                http://ads.bintangmawar.net/adlog.php? -
                DIRECT/63.243.152.34 image/gif
1178617445.202   4109 10.100.100.7 TCP_MISS/200 754 GET
                http://c17.statcounter.com/t.php? -
                DIRECT/207.44.158.88 image/png
1178617447.919   2715 10.100.100.7 TCP_MISS/200 1300 GET
                http://www.bintangmawar.net/i/nav_enter_o.gif -
                DIRECT/63.243.152.34 image/gif
```

Gambar 6.21 Isi Dari *File* /var/log/squid/access.log saat Mengakses www.bintangmawar.net

Alamat dari www.bintangmawar.net ditampilkan dalam Gambar 6.22.

Gambar 6.22 Firefox 2 menampilkan situs www.bintangmawar.net

Jika pengguna mengklik kotak putih di depan kalimat "`I have read and understand the agreement`" dan "`ENTER SITE`" maka pengguna akan mengakses ke alamat bintangmawar.net/forum. Pengaksesan tersebut akan tampak dalam *file* /var/log/squid/access.log dan /var/log/squid/parno.log. Isi dari *file* access.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1178618444.186      1 10.100.100.7 TCP_MISS/302 154 GET
                http://www.bintangmawar.net/forum/insideforumz.php
                - NONE/- -
```

Gambar 6.23 Isi Dari *File* /var/log/squid/access.log saat Mengakses www.bintangmawar.net/forum

Isi dari *file* parno.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f parno.log

2007-05-08 15:15:44 [31487] Request(lasif/parno/-)
                http://www.bintangmawar.net/forum/insideforumz.php
                10.100.100.7/- - GET REDIRECT
```

Gambar 6.24 Isi Dari *File* /var/log/squid/parno.log saat Mengakses www.bintangmawar.net/forum

Berdasarkan isi dari *file* access.log yang menunjukkan adanya keterangan "NONE" dan parno.log yang menunjukkan adanya keterangan "GET REDIRECT", dapat diketahui bahwa paket data dari komputer *client* saat melewati Squid diarahkan ke alamat lain. Paket data yang akan menuju bintangmawar.net/forum diarahkan ke alamat http://172.17.63.129:8080.

- Akses *file* mp3, avi, mpg dan 3gp

  Proses pen-*download*-an *file-file* mp3, avi, mpg dan 3gp dari alamat http://172.17.7.41:33444/wepe/index-nenggala.htm akan tampak dalam *file* /var/log/squid/access.log dan /var/log/squid/multimedia.log. Isi dari *file* access.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1178620851.735      1 10.100.100.7 TCP_MISS/302 154 GET
                http://172.17.7.41:33444/wepe/naruto.avi - NONE/-
                -
1178620853.768      1 10.100.100.7 TCP_MISS/302 154 GET
                http://172.17.7.41:33444/wepe/heroes.mpg - NONE/-
                -
1178620855.809      1 10.100.100.7 TCP_MISS/302 154 GET
                http://172.17.7.41:33444/wepe/life.mp3 - NONE/- -
1178620858.149      1 10.100.100.7 TCP_MISS/302 154 GET
                http://172.17.7.41:33444/wepe/handphone.3gp -
                NONE/- -
```

Gambar 6.25 Isi Dari *File* /var/log/squid/access.log saat Mengakses *File* mp3, avi, mpg dan 3gp

Isi dari *file* multimedia.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f multimedia.log

2007-05-08 15:23:51 [31742] Request(lasif/multimedia/-)
                    http://172.17.7.41:33444/wepe/naruto.avi
                    10.100.100.7/- - GET REDIRECT
2007-05-08 15:23:53 [31742] Request(lasif/multimedia/-)
                    http://172.17.7.41:33444/wepe/heroes.mpg
                    10.100.100.7/- - GET REDIRECT
2007-05-08 15:23:55 [31742] Request(lasif/multimedia/-)
                    http://172.17.7.41:33444/wepe/life.mp3
                    10.100.100.7/- - GET REDIRECT
2007-05-08 15:23:58 [31742] Request(lasif/multimedia/-)
                    http://172.17.7.41:33444/wepe/handphone.3gp
                    10.100.100.7/- - GET REDIRECT
```

Gambar 6.26 Isi Dari *File* /var/log/squid/multimedia.log saat Mengakses *File* mp3, avi, mpg dan 3gp

Berdasarkan isi dari *file* access.log yang menunjukkan adanya keterangan "NONE" dan multimedia.log yang menunjukkan adanya keterangan "GET REDIRECT", dapat diketahui bahwa paket data dari komputer *client* saat melewati Squid diarahkan ke alamat lain. Paket data yang akan menuju http://172.17.7.41:33444/wepe/naruto.avi, http://172.17.7.41:33444/wepe/heroes.mpg, http://172.17.7.41:33444/wepe/life.mp3, dan http://172.17.7.41:33444/wepe/handphone.3gp diarahkan ke alamat http://172.17.63.129:8080.

g. Kesimpulan

SquidGuard telah dapat bekerja sesuai dengan konfigurasi yang telah diberikan. SquidGuard tidak memblokir situs yang boleh diakses kapan saja seperti www.national-anime.com. SquidGuard dapat melakukan pemblokiran terhadap situs-situs yang tidak boleh diakses pada saat jam kerja dan membelokkan ke alamat http://172.17.63.129:8080. SquidGuard juga dapat melakukan pemblokiran saat pengguna men-*download file-file* mp3, avi, mpg, dan 3gp pada saat jam kerja dan membelokkan ke alamat http://172.17.63.129:8080.

6.5. Pengujian dan Analisis SquidGuard Secara Umum pada saat di luar Jam Kerja

Pengujian dan analisis konfigurasi SquidGuard secara umum pada saat di luar jam kerja, maksudnya adalah SquidGuard diuji langsung dengan komputer *client*. Dengan *browser* Mozilla Firefox 2, komputer *client* mengakses suatu situs dengan melewati SquidGuard. Langkah-langkah pengujian adalah sebagai berikut :

a. Tujuan

Pengujian ini dilakukan untuk mengetahui apakah SquidGuard telah berjalan sesuai dengan konfigurasi yang telah diberikan, saat digunakan dalam kondisi dan waktu secara umum pada saat di luar jam kerja. Kondisi dan waktu secara umum berarti SquidGuard memproses paket-paket data dari komputer *client* yang mengakses suatu situs dengan menggunakan *browser* [KOR-07].

b. Spesifikasi dan Konfigurasi Komputer

Komputer *client* dan komputer *gateway* memiliki spesifikasi sebagaimana yang telah disebutkan di bagian awal bab ini. Komputer *gateway* telah dikonfigurasi sebagaimana telah dijelaskan pada Bab Empat dan Bab Lima.

c. Software Aplikasi

- tcpdump 3.9.4
- Squid 2.6.STABLE4-1.fc6
- SquidGuard 1.2.0-14.fc6
- Mozilla Firefox 2
- tail (GNU coreutils) 5.97

d. Prosedur Pengujian

i. Mengaktifkan iptables, Squid dan SquidGuard di komputer *gateway*

Prosedur untuk mengaktifkan iptables dan Squid telah dijelaskan pada bagian "**6.3 Pengujian dan Analisis Squid**". SquidGuard merupakan *plug-in* dari Squid, sehingga saat Squid di aktifkan maka SquidGuard secara otomatis ikut aktif.

ii. Mengakses situs yang tidak diblokir pada saat di luar jam kerja

Situs tak terblokir yang akan diakses pada saat di luar jam kerja adalah www.national-anime.com. Situs tersebut dibuka dengan Firefox 2 di komputer *client*.

iii. Mengakses *web content* yang diblokir pada saat di luar jam kerja

Dalam prosedur ini, komputer *client* akan membuka beberapa *web content* yang diblokir pada saat jam kerja. *Web content* tersebut adalah sebagai berikut:

- bangbrosnetwork.com
- bintangmawar.net/forum
- *File* ber-*extention* mp3, avi, mpg, 3gp yang akan di-*download* dari http://172.17.7.41:33444/wepe/index-nenggala.htm.

Tampilan dari alamat http://172.17.7.41:33444/wepe/index-nenggala.htm diperlihatkan dalam Gambar 6.13.

e. Hasil Yang Diharapkan

Hasil yang diharapkan dari pengujian ini adalah pada saat di luar jam kerja, yaitu Senin sampai Jum'at pukul 15.31 hingga pukul 7.29 atau Hari Sabtu dan Minggu, Firefox 2 di komputer *client* dapat menampilkan situs www.national-anime.com, bangbrosnetwork.com, bintangmawar.net/forum, dan men-*download file* ber-*extention* mp3, avi, mpg, 3gp.

f. Hasil Pengujian dan Analisa

Hasil pengujian dan analisa dilakukan pada tiap tahap prosedur pengujian. Hasil pengujian dan analisa dijelaskan sebagaimana berikut :

i. Status SquidGuard

Untuk mengetahui apakah SquidGuard telah aktif dan siap memproses paket data, dapat dilakukan dengan cara melihat isi dari *file* /var/log/squid/squidGuard.log.

```
[root@Konoha squid]# tail -f squidGuard.log

2007-05-01 18:22:43 [17244] New setting: dbhome: /var/squidGuard/cream
2007-05-01 18:22:43 [17244] New setting: logdir: /var/log/squid
2007-05-01 18:22:43 [17244] init domainlist
                            /var/squidGuard/cream/parno-d
2007-05-01 18:22:43 [17244] loading dbfile
                            /var/squidGuard/cream/parno-d.db
2007-05-01 18:22:43 [17253] init urllist /var/squidGuard/cream/parno-u
2007-05-01 18:22:43 [17253] loading dbfile
                            /var/squidGuard/cream/parno-u.db
2007-05-01 18:22:43 [17254] init urllist /var/squidGuard/cream/parno-u
```

```
2007-05-01 18:22:43 [17254] loading dbfile
                            /var/squidGuard/cream/parno-u.db
2007-05-01 18:22:43 [17254] init expressionlist
                            /var/squidGuard/cream/media
2007-05-01 18:22:43 [17232] squidGuard 1.2.0 started (1178005017.821)
2007-05-01 18:22:43 [17232] recalculating alarm in 3243 seconds
2007-05-01 18:22:43 [17232] squidGuard ready for requests
                            (1178005017.918)
2007-05-01 18:22:43 [17238] init urllist /var/squidGuard/cream/parno-u
2007-05-01 18:22:43 [17238] loading dbfile
                            /var/squidGuard/cream/parno-u.db
2007-05-01 18:22:43 [17238] init expressionlist
                            /var/squidGuard/cream/media
2007-05-01 18:22:43 [17238] squidGuard 1.2.0 started (1178005017.857)
2007-05-01 18:22:43 [17238] recalculating alarm in 3243 seconds
2007-05-01 18:22:43 [17238] squidGuard ready for requests
                            (1178005017.920)
```

Gambar 6.27 Isi Dari *File* /var/log/squid/squidGuard.log

Di dalam file squidGuard.log, tampak bahwa "`squidGuard ready for requests`" yang berarti SquidGuard telah aktif dan siap memproses paket data.

ii. Akses situs yang tidak diblokir pada saat jam kerja

Dalam Gambar 6.16 tampak bahwa www.national-anime.com telah berhasil dibuka pada saat SquidGuard sedang aktif. Paket data komputer *client* yang mengakses www.national-anime.com tampak dalam *file* /var/log/squid/access.log. Isi dari *file* access.log saat membuka www.national-anime.com adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1178182004.353   7243 10.100.100.7 TCP_MISS/200 4739 GET
                 http://www.national-anime.com/ -
                 DIRECT/208.122.8.190 text/html
1178182007.231   4511 10.100.100.7 TCP_MISS/200 2089 GET
                 http://www.national-anime.com/all.css -
                 DIRECT/208.122.8.190 text/css
1178182008.543   9389 10.100.100.7 TCP_MISS/200 582 GET
                 http://www.national-anime.com/images/menu_bg.gif -
                 DIRECT/208.122.8.190 image/gif
1178182010.531   1245 10.100.100.7 TCP_MISS/200 580 GET
                 http://www.national-anime.com/images/body-
                 background.gif - DIRECT/208.122.8.190 image/gif
1178182015.173  45644 10.100.100.7 TCP_MISS/200 51807 GET
                 http://www.national-anime.com/images/banner.jpg -
                 DIRECT/208.122.8.190 image/jpeg
1178182022.134  56123 10.100.100.7 TCP_MISS/302 625 GET
                 http://www.national-anime.com/favicon.ico -
                 DIRECT/208.122.8.190 text/html
```

```
1178182031.865  11223 10.100.100.7 TCP_MISS/200 5956 GET
                http://sb.google.com/safebrowsing/update? -
                DIRECT/72.14.253.93 text/plain
```

Gambar 6.28 Isi Dari *File* /var/log/squid/access.log pada Pengujian SquidGuard secara Umum

Dari isi *file* access.log, dapat disimpulkan bahwa paket data dari IP 10.100.100.7 pergi menuju IP 208.122.8.190 dengan perantara Squid. Hasil pengaksesan situs tersebut ditunjukan dalam Gambar 6.16.

iii. Akses *web content* yang diblokir pada saat jam kerja

- Akses alamat bangbrosnetwork.com

Akses ke alamat bangbrosnetwork.com yang dilakukan oleh komputer *client* akan tampak dalam *file* /var/log/squid/access.log. Isi dari *file* access.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1190262549.123   1413 10.100.100.7 TCP_MISS/200 8463 GET
                 http://www.bangbrosnetwork.com/ -
                 DIRECT/66.230.129.242 text/html
1190262549.729   288 10.100.100.7 TCP_MISS/200 2272 GET
                 http://xml.alexa.com/data? -
                 DIRECT/64.213.200.101 text/xml
1190262551.254   50 10.100.100.7 TCP_HIT/200 1723 GET
                 http://images1.bangbros.com/shared/icra.gif -
                 NONE/- image/gif
1190262552.830   2928 10.100.100.7 TCP_MISS/200 24408 GET
                 http://images2.bangbros.com/bangbrosnetwork/t1/2
                 .jpg - DIRECT/66.230.128.210
                 image/jpeg
.
.
.
1190264409.528   10687 10.100.100.7 TCP_HIT/200 57820 GET
                 http://images2.bangbros.com/bangbrosnetwork/t1/2
                 .jpg - NONE/- image/jpeg
1190264611.969   529 10.100.100.7 TCP_MISS/200 213 GET
                 http://sb.google.com/safebrowsing/update? -
                 DIRECT/209.85.163.91 text/html
```

Gambar 6.29 Isi Dari *File* /var/log/squid/access.log saat Mengakses www.bangbrosnetwork.com

Berdasarkan isi dari *file* access.log, dapat disimpulkan bahwa paket data dari IP 10.100.100.7 pergi menuju IP 66.230.188.2 dengan perantara Squid. Hasil pengaksesan situs tersebut ditunjukan dalam Gambar 6.30.

Gambar 6.30 Firefox 2 menampilkan situs www.bangbrosnetwork.com

- Akses alamat bintangmawar.net/forum

  Saat komputer *client* mengakses alamat www.bintangmawar.net, Firefox 2 berhasil menampilkan isi dari alamat tersebut. Isi dari *file* access.log saat membuka www.bintangmawar.net ditunjukkan dalam Gambar 6.22.

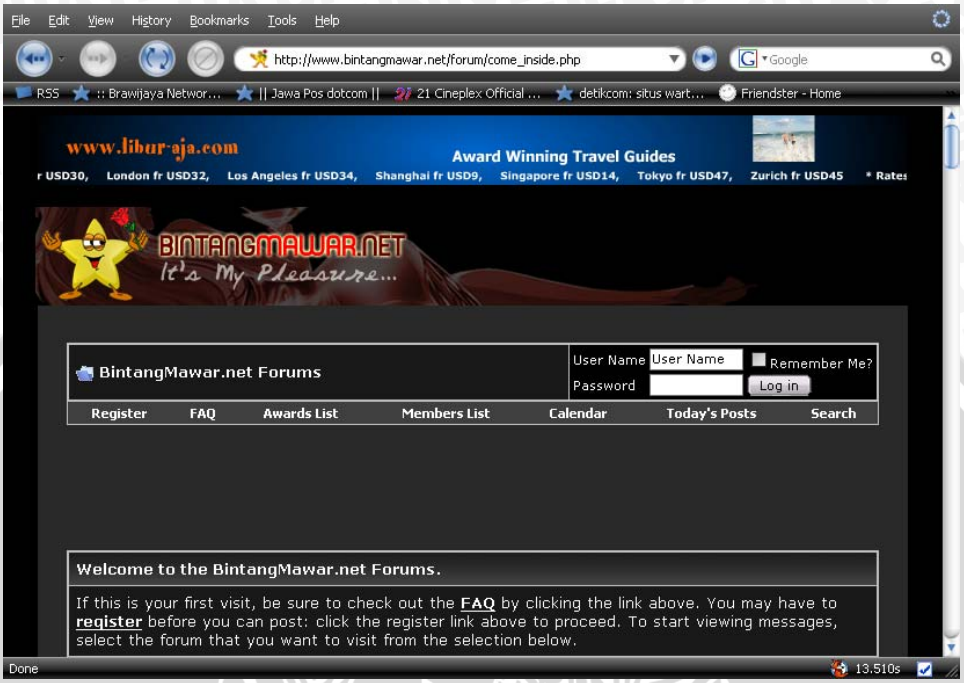
  Jika pengguna mengklik kotak putih di depan kalimat "I have read and understand the agreement" dan "ENTER SITE" maka pengguna akan mengakses ke alamat bintangmawar.net/forum. Pengaksesan tersebut akan tampak dalam *file* /var/log/squid/access.log. Isi dari *file* access.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1190347281.670  289 10.100.100.7 TCP_MISS/200 4091 GET
                http://www.bintangmawar.net/forum/favicon.ico -
                DIRECT/63.243.152.34 image/x-icon
1190347286.077  14239 10.100.100.7 TCP_MISS/200 19231 GET
                http://www.bintangmawar.net/forum/come_inside.php
                - DIRECT/63.243.152.34 text/html
.
.
1178618444.186  1 10.100.100.7 TCP_MISS/302 154 GET
                http://www.bintangmawar.net/forum/insideforumz.php
                - NONE/- -
```

Gambar 6.31 Isi Dari *File* /var/log/squid/access.log saat Mengakses www.bintangmawar.net/forum

Berdasarkan isi dari *file* access.log, dapat disimpulkan bahwa paket data dari IP 10.100.100.7 pergi menuju IP www.bintangmawar.net/forum dengan perantara Squid. Hasil pengaksesan situs tersebut ditunjukan dalam Gambar 6.32.



Gambar 6.32 Firefox 2 menampilkan situs www.bintangmawar.net/forum

- Akses *file* mp3, avi, mpg dan 3gp

  Proses pen-*download*-an *file-file* mp3, avi, mpg dan 3gp dari alamat http://172.17.7.41:33444/wepe/index-nenggala.htm akan tampak dalam *file* /var/log/squid/access.log dan. Isi dari *file* access.log adalah sebagai berikut :

```
[root@Konoha squid]# tail -f access.log

1190355644.368   347 10.100.100.7 TCP_MISS/200 421 GET
                 http://172.17.7.41:33444/wepe/naruto.avi -
                 DIRECT/172.17.7.41 video/x-msvideo
1190355649.726   362 10.100.100.7 TCP_MISS/200 416 GET
                 http://172.17.7.41:33444/wepe/heroes.mpg -
                 DIRECT/172.17.7.41 video/mpeg
1190355654.365  1216 10.100.100.7 TCP_MISS/200 5904981 GET
                 http://172.17.7.41:33444/wepe/life.mp3 -
                 DIRECT/172.17.7.41 audio/mpeg
1190355656.407   43 10.100.100.7 TCP_MISS/200 416 GET
                 http://172.17.7.41:33444/wepe/handphone.3gp -
                 DIRECT/172.17.7.41 text/plain
```

Gambar 6.33 Isi Dari *File* /var/log/squid/access.log saat Mengakses *File* mp3, avi, mpg dan 3gp

Berdasarkan isi dari *file* access.log, dapat disimpulkan bahwa IP 10.100.100.7 dapat men--*download*-an *file-file* mp3, avi, mpg dan 3gp dari alamat http://172.17.7.41:33444/wepe/index-nenggala.htm.

g. Kesimpulan

SquidGuard telah dapat bekerja sesuai dengan konfigurasi yang telah diberikan. Saat di luar jam kerja, , yaitu Senin sampai Jum'at pukul 15.31 hingga pukul 7.29 atau Hari Sabtu dan Minggu, SquidGuard tidak melakukan pemblokiran.

BAB VII

PENUTUP

7.1. Kesimpulan

Dari hasil pengujian dan analisis sistem pemfilteran layanan Internet pada *gateway* dengan menggunakan SquidGuard, dapat disimpulkan bahwa :

1. Komputer *gateway* digunakan untuk menghubungkan dua *network* yang berlainan, yaitu *internal network* dan *external network* sehingga paket data dari *internal network* yang dibuat oleh komputer *client* dapat diteruskan menuju *external network.*
2. Squid dapat menjalankan SquidGuard dan memproses paket data yang melaluinya dengan baik tanpa berfungsi sebagai *web cache.*
3. SquidGuard dapat memfilter paket data yang terlarang pada saat yang terlarang dan me-*redirect* paket data tersebut ke alamat lain. Pada saat yang sama, SquidGuard membiarkan paket data yang tidak terlarang pada saat terlarang untuk tetap berjalan menuju *external network.*

7.2. Saran

Saran yang dapat diberikan untuk pengembangan sistem ini adalah penambahan suatu sistem penganalisa *log* yang berbasis grafis, dapat bekerja secara *realtime*, sekaligus dapat membuat sebuah laporan.

DAFTAR PUSTAKA

[ALM-01] Almesberger, Werner. 2001. *Linux Networking Traffic Control - Implementation Overview*. EPFL ICA.

[BAL-02] Balzertsen, Päl dan Lars Eik Häland. 2002. *The squidGuard Introduction*. Aarhus : Tele Danmark InterNordia. Akses dari : http://www.squidguard.org/intro/index.html. Tanggal akses : 11 Oktober 2006

[BAR-07] Barron, Daniel. 2007. *DansGuardian - Introduction*. Akses dari : http://dansguardian.org/?page=introduction. Tanggal akses : 2 Agustus 2007

[CAL-02] Caldera International Inc. 2002. *The TCP/IP protocol stack*. Akses dari : http://uw713doc.sco.com/en/NET_tcpip/tcpN.tcpip_stack.html. Tanggal akses : 11 Oktober 2006

[CHA-96] Chatel, M. 1996. *RFC 1919 - Classical versus Transparent IP Proxies*. Akses dari : http://www.ietf.org/rfc/rfc1919.txt?number=1919. Tanggal akses : 28 Agustus 2007

[CIS-04] *Anonymous*. 2004. *CCNA 1 v3.1 Module 9 TCP/IP Protocol Suite and IP Addressing*. Cisco Systems Inc.

[CON-05a] Conniq.com. 2005. *Networking Guide : Network Topologies*. Akses dari : http://www.conniq.com/Networking_Topology.htm. Tanggal akses : 13 Oktober 2006

[CON-05b] Conniq.com. 2005. *Networking Guide : Network Topologies*. Akses dari : http://www.conniq.com/Networking_Topology3.htm. Tanggal akses : 13 Oktober 2006

[EPI-05] Epipe Networks Pty. Ltd. 2005. *Glossary of Terms*. Akses dari : http://www.ml-ip.com/html/support/glossary.html#R. Tanggal akses : 26 Agustus 2007

[FLO-05] Florida Center for Instructional Technology College of Education. 2005. *Chapter 5 : Topology*. Akses dari : http://fcit.usf.edu/network/chap5/chap5.htm. Tanggal akses : 13 Oktober 2006

[IES-03] Institute of Education Sciences. 2003. *Weaving a Secure Web Around Education : A Guide to Technology Standards and Security - Glossary*. Akses dari : http://nces.ed.gov/pubs2003/secureweb/glossary.asp . Tanggal akses : 16 Agustus 2007

[ISI-81a] Information Sciences Institute. 1981. *RFC 793 - Transmission Control Protocol*. Akses dari : http://www.networksorcery.com/enp/rfc/rfc793.txt. Tanggal akses : 14 Mei 2007

[ISI-81b] Information Sciences Institute. 1981. *RFC 791 - Internet Protocol*. Akses dari : http://www.networksorcery.com/enp/rfc/rfc793.txt. Tanggal akses : 14 Mei 2007

[INT-99] Internet Society. 1999. *RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1*. Akses dari http://www.ietf.org/rfc/rfc2616.txt. Tanggal akses : 14 Mei 2007

[KOR-07] Kronberg, Christine. 2007. *Basic Configuration of SquidGuard*. Akses dari : http://www.squidguard.org/Doc/configure.html. Tanggal akses : 26 Agustus 2007.

[MAL-02] Malhotra, Ravi. 2002. *IP Routing*. Sebastopol : O'Reilly & Associates, Inc.

[NCT-07] Netfilter Core Team. 2007. *The netfilter.org "iptables" project*. Akses dari : http://www.netfilter.org/projects/iptables/index.html. Tanggal akses : 14 Mei 2007

[POS-80:1] Postel, J. 1980. *RFC 768 - User Datagram Protocol*. Akses dari : http://tools.ietf.org/html/rfc768. Tanggal akses : 14 Mei 2007

[PUR-01] Purbo, Onno W., et al., 2001. *TCP/IP : Standar, Desain, dan Implementasi*. Jakarta : PT Elex Media Komputindo.

[SUG-07] Sugeng Purwantoro E.S.G.S. 2007. *Konfigurasi Dasar PC-Router dengan Windows 2003 Server*. Akses dari : http://ilmukomputer.com/2007/03/03/konfigurasi-dasar-pc-router-dengan-windows-2003-server/. Tanggal akses : 16 Agustus 2007

[TEA-04] Team Squid. 2004. *SQUID Frequently Asked Questions*. Akses dari : http://www.squid-cache.org/Doc/FAQ/FAQ.html. Tanggal akses : 11 Oktober 2006

[TEX-07] Texas State Library and Archives Commission. 2007. *Glossary*. Akses dari : http://www.tsl.state.tx.us/ld/pubs/compsecurity/glossary.html. Tanggal akses : 26 Agustus 2007.

[URL-07] URLfilterDB B.V. 2007. *URLfilterDB - Home*. Akses dari : http://www.urlfilterdb.com/index.shtml. Tanggal akses : 2 Agustus 2007

[VAS-07] Vaskovich, Gordon Lyon. 2007. *Nmap - Free Security Scanner For Network Exploration & Security Audits – Introduction*. Akses dari : http://insecure.org/nmap/. Tanggal akses : 20 September 2007

[WIK-06] Wikipedia.org. 2006. *Image:Iptables packet traversal.pdf*. Akses dari : http://en.wikipedia.org/wiki/Image:Iptables_packet_traversal.pdf. Tanggal akses : 29 November 2006

[WIL-93] Wilder, Floyd. 1993. *A Guide To The TCP/IP Protocol Suite*. Artech House, Inc: Norwood, MA.

LAMPIRAN

LAMPIRAN 1

Hasil tcpdump Dari Komputer *Client* Menuju *Server* www.klikbca.com

```
[root@Konoha ~]# tcpdump -ni eth2 ip src 10.100.100.7 and dst port 80

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes

23:30:05.471937 IP 10.100.100.7.36200 > 202.6.211.8.http: S 3187407278:3187407278(0)
        win 5840 <mss 1460,sackOK,timestamp 9329149 0,nop,wscale 5>
23:30:05.473976 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 585278218 win 183
        <nop,nop,timestamp 9329149 130861582>
23:30:05.475723 IP 10.100.100.7.36200 > 202.6.211.8.http: P 0:399(399) ack 1 win 183
        <nop,nop,timestamp 9329150 130861582>
23:30:06.403265 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 1390 win 273
        <nop,nop,timestamp 9329382 130862507>
23:30:06.413944 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 2758 win 360
        <nop,nop,timestamp 9329384 130862517>
23:30:06.461333 IP 10.100.100.7.36201 > 202.6.211.8.http: S 3182956103:3182956103(0)
        win 5840 <mss 1460,sackOK,timestamp 9329396 0,nop,wscale 5>
23:30:06.464516 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 587995201 win 183
        <nop,nop,timestamp 9329397 130862572>
23:30:06.465272 IP 10.100.100.7.36201 > 202.6.211.8.http: P 0:434(434) ack 1 win 183
        <nop,nop,timestamp 9329397 130862572>
23:30:06.471801 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 1449 win 273
        <nop,nop,timestamp 9329399 130862577>
23:30:06.987730 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 2897 win 364
        <nop,nop,timestamp 9329528 130862577>
23:30:06.987766 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 3046 win 454
        <nop,nop,timestamp 9329528 130862582>
23:30:06.993769 IP 10.100.100.7.36201 > 202.6.211.8.http: P 434:855(421) ack 3046 win
        454 <nop,nop,timestamp 9329529 130862582>
23:30:07.195133 IP 10.100.100.7.36201 > 202.6.211.8.http: P 434:855(421) ack 3046 win
        454 <nop,nop,timestamp 9329580 130862582>
23:30:07.331711 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 3046 win 454
        <nop,nop,timestamp 9329614 130862582,nop,nop,sack 1 {1449:2897}>
23:30:07.331988 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 4494 win 545
        <nop,nop,timestamp 9329614 130863107>
23:30:07.936437 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 5942 win 635
        <nop,nop,timestamp 9329765 130863107>
23:30:07.937052 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 7094 win 726
        <nop,nop,timestamp 9329765 130863107>
23:30:08.481929 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 4126 win 446
        <nop,nop,timestamp 9329901 130863138>
23:30:08.481998 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 5494 win 531
        <nop,nop,timestamp 9329901 130863146>
23:30:09.053647 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 5494 win 531
        <nop,nop,timestamp 9330044 130863146,nop,nop,sack 1 {2758:4126}>
23:30:09.054902 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 8542 win 816
        <nop,nop,timestamp 9330044 130863446>
23:30:09.656437 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 9990 win 907
        <nop,nop,timestamp 9330195 130863446>
23:30:09.657317 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 5494 win 531
        <nop,nop,timestamp 9330195 130863146,nop,nop,sack 1 {2758:4126}>
23:30:10.245328 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 11438 win 997
        <nop,nop,timestamp 9330342 130864051>
23:30:10.245476 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 12886 win 1088
        <nop,nop,timestamp 9330342 130864051>
23:30:10.849987 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 14334 win 1178
        <nop,nop,timestamp 9330493 130864051>
23:30:10.850385 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 15286 win 1269
        <nop,nop,timestamp 9330493 130864051>
23:30:11.355774 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 5494 win 531
        <nop,nop,timestamp 9330620 130863146,nop,nop,sack 1 {2758:4126}>
23:30:11.355840 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 5494 win 531
        <nop,nop,timestamp 9330620 130864594,nop,nop,sack 1 {4126:5494}>
23:30:11.927536 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 6942 win 622
        <nop,nop,timestamp 9330763 130864594>
23:30:11.927586 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 7137 win 712
        <nop,nop,timestamp 9330763 130864594>
```

```
23:30:12.283922 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 16734 win 1359
        <nop,nop,timestamp 9330852 130865167>
23:30:12.283956 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 18182 win 1450
        <nop,nop,timestamp 9330852 130865167>
23:30:12.886046 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 19630 win 1540
        <nop,nop,timestamp 9331002 130865770>
23:30:12.887266 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 21078 win 1631
        <nop,nop,timestamp 9331003 130865770>
23:30:13.490628 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 22526 win 1721
        <nop,nop,timestamp 9331153 130866358>
23:30:13.491330 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 23478 win 1812
        <nop,nop,timestamp 9331154 130866358>
23:30:13.995467 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 24926 win 1902
        <nop,nop,timestamp 9331280 130866358>
23:30:13.996612 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 26374 win 1993
        <nop,nop,timestamp 9331280 130866358>
23:30:14.600334 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 27822 win 2003
        <nop,nop,timestamp 9331431 130866963>
23:30:14.600366 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 29270 win 1962
        <nop,nop,timestamp 9331431 130866963>
23:30:15.206451 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 30718 win 2003
        <nop,nop,timestamp 9331582 130866965>
23:30:15.206517 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 31670 win 1973
        <nop,nop,timestamp 9331582 130866965>
23:30:15.709685 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 7137 win 712
        <nop,nop,timestamp 9331708 130864594,nop,nop,sack 1 {5494:6942}>
23:30:15.710705 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 33118 win 2003
        <nop,nop,timestamp 9331708 130868397>
23:30:16.314821 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 34133 win 2003
        <nop,nop,timestamp 9331859 130868397>
23:30:16.338218 IP 10.100.100.7.36200 > 202.6.211.8.http: P 399:819(420) ack 7137 win
        712 <nop,nop,timestamp 9331865 130864594>
23:30:16.343686 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 8585 win 803
        <nop,nop,timestamp 9331867 130872455>
23:30:16.845145 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 10033 win 893
        <nop,nop,timestamp 9331992 130872455>
23:30:16.845853 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 11186 win 984
        <nop,nop,timestamp 9331992 130872455>
23:30:17.391971 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 12634 win 1074
        <nop,nop,timestamp 9332129 130872459>
23:30:17.392006 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 14082 win 1165
        <nop,nop,timestamp 9332129 130872962>
23:30:17.996368 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 15429 win 1255
        <nop,nop,timestamp 9332280 130872962>
23:30:18.191683 IP 10.100.100.7.36200 > 202.6.211.8.http: P 814:1253(434) ack 15429
        win 1255 <nop,nop,timestamp 9332329 130872962>
23:30:18.191733 IP 10.100.100.7.36201 > 202.6.211.8.http: P 855:1289(434) ack 34133
        win 2003 <nop,nop,timestamp 9332329 130868397>
23:30:18.191953 IP 10.100.100.7.36202 > 202.6.211.8.http: S 3202392590:3202392590(0)
        win 5840 <mss 1460,sackOK,timestamp 9332329 0,nop,wscale 5>
23:30:18.192006 IP 10.100.100.7.36203 > 202.6.211.8.http: S 3202746616:3202746616(0)
        win 5840 <mss 1460,sackOK,timestamp 9332329 0,nop,wscale 5>
23:30:18.196330 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 597004165 win 183
        <nop,nop,timestamp 9332330 130874309>
23:30:18.196366 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 591197558 win 183
        <nop,nop,timestamp 9332330 130874309>
23:30:18.199663 IP 10.100.100.7.36203 > 202.6.211.8.http: P 0:440(440) ack 1 win 183
        <nop,nop,timestamp 9332331 130874309>
23:30:18.199712 IP 10.100.100.7.36202 > 202.6.211.8.http: P 0:437(437) ack 1 win 183
        <nop,nop,timestamp 9332331 130874309>
23:30:18.199774 IP 10.100.100.7.36204 > 202.6.211.8.http: S 3208415529:3208415529(0)
        win 5840 <mss 1460,sackOK,timestamp 9332331 0,nop,wscale 5>
23:30:18.199828 IP 10.100.100.7.36205 > 202.6.211.8.http: S 3201165971:3201165971(0)
        win 5840 <mss 1460,sackOK,timestamp 9332331 0,nop,wscale 5>
23:30:18.199928 IP 10.100.100.7.36206 > 202.6.211.8.http: S 3205423461:3205423461(0)
        win 5840 <mss 1460,sackOK,timestamp 9332331 0,nop,wscale 5>
23:30:18.200318 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 35581 win 2003
        <nop,nop,timestamp 9332331 130874310>
23:30:18.207823 IP 10.100.100.7.36207 > 202.6.211.8.http: S 3208163835:3208163835(0)
        win 5840 <mss 1460,sackOK,timestamp 9332333 0,nop,wscale 5>
23:30:18.395453 IP 10.100.100.7.36200 > 202.6.211.8.http: P 814:1253(434) ack 15429
        win 1255 <nop,nop,timestamp 9332380 130872962>
23:30:18.403443 IP 10.100.100.7.36203 > 202.6.211.8.http: P 0:440(440) ack 1 win 183
        <nop,nop,timestamp 9332382 130874309>
23:30:18.403490 IP 10.100.100.7.36202 > 202.6.211.8.http: P 0:437(437) ack 1 win 183
        <nop,nop,timestamp 9332382 130874309>
```

```
23:30:18.620224 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 37029 win 2003
        <nop,nop,timestamp 9332436 130874310>
23:30:18.620291 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 37917 win 1975
        <nop,nop,timestamp 9332436 130874310>
23:30:18.620767 IP 10.100.100.7.36201 > 202.6.211.8.http: P 1289:1728(439) ack 37917
        win 2003 <nop,nop,timestamp 9332436 130874310>
23:30:18.803389 IP 10.100.100.7.36200 > 202.6.211.8.http: P 814:1253(434) ack 15429
        win 1255 <nop,nop,timestamp 9332482 130872962>
23:30:18.811316 IP 10.100.100.7.36203 > 202.6.211.8.http: P 0:440(440) ack 1 win 183
        <nop,nop,timestamp 9332484 130874309>
23:30:18.811371 IP 10.100.100.7.36202 > 202.6.211.8.http: P 0:437(437) ack 1 win 183
        <nop,nop,timestamp 9332484 130874309>
23:30:18.995333 IP 10.100.100.7.36201 > 202.6.211.8.http: P 1289:1728(439) ack 37917
        win 2003 <nop,nop,timestamp 9332530 130874310>
23:30:19.111115 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 595130555 win 183
        <nop,nop,timestamp 9332558 130874318>
23:30:19.111145 IP 10.100.100.7.36205 > 202.6.211.8.http: . ack 597494179 win 183
        <nop,nop,timestamp 9332558 130874318>
23:30:19.111176 IP 10.100.100.7.36206 > 202.6.211.8.http: . ack 603847601 win 183
        <nop,nop,timestamp 9332558 130874318>
23:30:19.111571 IP 10.100.100.7.36206 > 202.6.211.8.http: P 0:438(438) ack 1 win 183
        <nop,nop,timestamp 9332559 130874318>
23:30:19.111616 IP 10.100.100.7.36205 > 202.6.211.8.http: P 0:438(438) ack 1 win 183
        <nop,nop,timestamp 9332559 130874318>
23:30:19.111655 IP 10.100.100.7.36204 > 202.6.211.8.http: P 0:438(438) ack 1 win 183
        <nop,nop,timestamp 9332559 130874318>
23:30:19.113982 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 1449 win 273
        <nop,nop,timestamp 9332559 130874319>
23:30:19.483094 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 2344 win 364
        <nop,nop,timestamp 9332651 130874319>
23:30:19.483644 IP 10.100.100.7.36202 > 202.6.211.8.http: P 437:877(440) ack 2344 win
        364 <nop,nop,timestamp 9332652 130874319>
23:30:19.484940 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 1449 win 273
        <nop,nop,timestamp 9332652 130874319>
23:30:19.619413 IP 10.100.100.7.36200 > 202.6.211.8.http: P 814:1253(434) ack 15429
        win 1255 <nop,nop,timestamp 9332686 130872962>
23:30:19.747355 IP 10.100.100.7.36201 > 202.6.211.8.http: P 1289:1728(439) ack 37917
        win 2003 <nop,nop,timestamp 9332718 130874310>
23:30:19.978774 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 2897 win 364
        <nop,nop,timestamp 9332775 130874319>
23:30:19.978857 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 595105993 win 183
        <nop,nop,timestamp 9332775 130874330>
23:30:19.979862 IP 10.100.100.7.36207 > 202.6.211.8.http: P 0:439(439) ack 1 win 183
        <nop,nop,timestamp 9332776 130874330>
23:30:20.308228 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 2344 win 364
        <nop,nop,timestamp 9332858 130874522,nop,nop,sack 1 {1:1449}>
23:30:20.511432 IP 10.100.100.7.36202 > 202.6.211.8.http: P 437:877(440) ack 2344 win
        364 <nop,nop,timestamp 9332909 130874522>
23:30:20.654336 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 2897 win 364
        <nop,nop,timestamp 9332944 130874521,nop,nop,sack 1 {1:1449}>
23:30:20.654935 IP 10.100.100.7.36201 > 202.6.211.8.http: P 1728:2164(436) ack 39035
        win 2003 <nop,nop,timestamp 9332944 130874740>
23:30:21.192027 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 16877 win 1346
        <nop,nop,timestamp 9333079 130874753>
23:30:21.193168 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 18325 win 1436
        <nop,nop,timestamp 9333079 130874753>
23:30:21.796415 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 19569 win 1527
        <nop,nop,timestamp 9333230 130874753>
23:30:21.797860 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 21017 win 1617
        <nop,nop,timestamp 9333230 130874753>
23:30:21.835432 IP 10.100.100.7.36206 > 202.6.211.8.http: P 0:438(438) ack 1 win 183
        <nop,nop,timestamp 9333240 130874318>
23:30:21.835494 IP 10.100.100.7.36205 > 202.6.211.8.http: P 0:438(438) ack 1 win 183
        <nop,nop,timestamp 9333240 130874318>
23:30:21.835516 IP 10.100.100.7.36204 > 202.6.211.8.http: P 0:438(438) ack 1 win 183
        <nop,nop,timestamp 9333240 130874318>
23:30:22.360531 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 2344 win 364
        <nop,nop,timestamp 9333371 130874930,nop,nop,sack 1 {1:1449}>
23:30:22.567517 IP 10.100.100.7.36202 > 202.6.211.8.http: P 437:877(440) ack 2344 win
        364 <nop,nop,timestamp 9333423 130874930>
23:30:22.706077 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 2897 win 364
        <nop,nop,timestamp 9333457 130874929,nop,nop,sack 1 {1:1449}>
23:30:23.034139 IP 10.100.100.7.36206 > 202.6.211.8.http: . ack 1449 win 273
        <nop,nop,timestamp 9333539 130875230>
23:30:23.034172 IP 10.100.100.7.36206 > 202.6.211.8.http: . ack 1990 win 364
        <nop,nop,timestamp 9333539 130875230>
```

```
23:30:23.034827 IP 10.100.100.7.36206 > 202.6.211.8.http: P 438:873(435) ack 1990 win
        364 <nop,nop,timestamp 9333539 130875230>
23:30:23.159465 IP 10.100.100.7.36201 > 202.6.211.8.http: P 1728:2164(436) ack 39035
        win 2003 <nop,nop,timestamp 9333571 130875114>
23:30:23.470098 IP 10.100.100.7.36205 > 202.6.211.8.http: . ack 1449 win 273
        <nop,nop,timestamp 9333648 130875230>
23:30:23.470132 IP 10.100.100.7.36205 > 202.6.211.8.http: . ack 1933 win 364
        <nop,nop,timestamp 9333648 130875230>
23:30:23.470821 IP 10.100.100.7.36205 > 202.6.211.8.http: P 438:873(435) ack 1933 win
        364 <nop,nop,timestamp 9333648 130875230>
23:30:23.882292 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 2344 win 364
        <nop,nop,timestamp 9333751 130875235,nop,nop,sack 1 {1449:2344}>
23:30:23.883657 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 1092 win 251
        <nop,nop,timestamp 9333752 130875235>
23:30:23.884264 IP 10.100.100.7.36204 > 202.6.211.8.http: P 438:877(439) ack 1092 win
        251 <nop,nop,timestamp 9333752 130875235>
23:30:24.315501 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 3792 win 454
        <nop,nop,timestamp 9333860 130875602>
23:30:24.316687 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 5240 win 545
        <nop,nop,timestamp 9333860 130875602>
23:30:24.934344 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 2897 win 364
        <nop,nop,timestamp 9334014 130875604,nop,nop,sack 1 {1449:2897}>
23:30:24.934408 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 4345 win 454
        <nop,nop,timestamp 9334014 130875604>
23:30:25.283526 IP 10.100.100.7.36207 > 202.6.211.8.http: P 0:439(439) ack 1 win 183
        <nop,nop,timestamp 9334102 130874330>
23:30:25.536914 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 5793 win 545
        <nop,nop,timestamp 9334165 130876097>
23:30:25.868758 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 7241 win 635
        <nop,nop,timestamp 9334248 130876097>
23:30:25.869947 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 1449 win 273
        <nop,nop,timestamp 9334248 130876100>
23:30:26.484658 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 2228 win 364
        <nop,nop,timestamp 9334402 130876100>
23:30:26.485246 IP 10.100.100.7.36207 > 202.6.211.8.http: P 439:872(433) ack 2228 win
        364 <nop,nop,timestamp 9334402 130876100>
23:30:26.523483 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 40483 win 2003
        <nop,nop,timestamp 9334412 130876774>
23:30:26.981753 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 40842 win 2003
        <nop,nop,timestamp 9334526 130876774>
23:30:26.982266 IP 10.100.100.7.36201 > 202.6.211.8.http: P 2164:2597(433) ack 40842
        win 2003 <nop,nop,timestamp 9334526 130876774>
23:30:26.984392 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 22465 win 1708
        <nop,nop,timestamp 9334527 130877310>
23:30:27.370225 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 23913 win 1798
        <nop,nop,timestamp 9334623 130877312>
23:30:27.370260 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 5240 win 545
        <nop,nop,timestamp 9334623 130876631,nop,nop,sack 1 {2344:3792}>
23:30:27.975845 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 25361 win 1889
        <nop,nop,timestamp 9334775 130877916>
23:30:27.975908 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 26809 win 1962
        <nop,nop,timestamp 9334775 130877917>
23:30:28.319618 IP 10.100.100.7.36206 > 202.6.211.8.http: P 438:873(435) ack 1990 win
        364 <nop,nop,timestamp 9334861 130875230>
23:30:28.580553 IP 10.100.100.7.36205 > 202.6.211.8.http: . ack 1933 win 364
        <nop,nop,timestamp 9334926 130877955,nop,nop,sack 1 {1:1449}>
23:30:28.924354 IP 10.100.100.7.36206 > 202.6.211.8.http: . ack 1990 win 364
        <nop,nop,timestamp 9335012 130877954,nop,nop,sack 1 {1:1449}>
23:30:28.924925 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 1092 win 251
        <nop,nop,timestamp 9335012 130877965,nop,nop,sack 1 {1:1092}>
23:30:29.127663 IP 10.100.100.7.36205 > 202.6.211.8.http: P 438:873(435) ack 1933 win
        364 <nop,nop,timestamp 9335063 130877955>
23:30:29.457033 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 7241 win 635
        <nop,nop,timestamp 9335145 130876097,nop,nop,sack 1 {2897:4345}>
23:30:29.786706 IP 10.100.100.7.36206 > 202.6.211.8.http: . ack 3198 win 454
        <nop,nop,timestamp 9335227 130879156>
23:30:29.787090 IP 10.100.100.7.36205 > 202.6.211.8.http: . ack 2767 win 454
        <nop,nop,timestamp 9335227 130879592>
23:30:29.787693 IP 10.100.100.7.36206 > 202.6.211.8.http: P 873:1314(441) ack 3198 win
        454 <nop,nop,timestamp 9335228 130879156>
23:30:30.246683 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 2503 win 342
        <nop,nop,timestamp 9335342 130880047>
23:30:30.246867 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 6688 win 635
        <nop,nop,timestamp 9335342 130880435>
23:30:30.857454 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 8136 win 726
        <nop,nop,timestamp 9335495 130880435>
```

```
23:30:30.857521 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 9584 win 816
        <nop,nop,timestamp 9335495 130880437>
23:30:31.462241 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 8689 win 726
        <nop,nop,timestamp 9335646 130881054>
23:30:31.462273 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 2228 win 364
        <nop,nop,timestamp 9335646 130876100,nop,nop,sack 1 {1:1449}>
23:30:32.065783 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 10137 win 816
        <nop,nop,timestamp 9335797 130881991>
23:30:32.395517 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 3676 win 454
        <nop,nop,timestamp 9335879 130882607>
23:30:32.395624 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 5124 win 545
        <nop,nop,timestamp 9335880 130882607>
23:30:32.997755 IP 10.100.100.7.36207 > 202.6.211.8.http: . ack 5433 win 635
        <nop,nop,timestamp 9336030 130882607>
23:30:33.035670 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 42290 win 2003
        <nop,nop,timestamp 9336040 130883105>
23:30:33.388259 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 43738 win 2003
        <nop,nop,timestamp 9336128 130883105>
23:30:33.388382 IP 10.100.100.7.36201 > 202.6.211.8.http: . ack 44530 win 1978
        <nop,nop,timestamp 9336128 130883105>
23:30:33.861077 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 28257 win 2003
        <nop,nop,timestamp 9336246 130883108>
23:30:33.861108 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 29705 win 1962
        <nop,nop,timestamp 9336246 130883108>
23:30:34.465623 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 31153 win 2003
        <nop,nop,timestamp 9336397 130883491>
23:30:34.465759 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 31857 win 1981
        <nop,nop,timestamp 9336397 130884097>
23:30:34.921436 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 33305 win 2003
        <nop,nop,timestamp 9336511 130884097>
23:30:35.252945 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 10690 win 907
        <nop,nop,timestamp 9336594 130886370>
23:30:35.287668 IP 10.100.100.7.36206 > 202.6.211.8.http: . ack 4150 win 545
        <nop,nop,timestamp 9336603 130885912>
23:30:35.702193 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 11585 win 907
        <nop,nop,timestamp 9336706 130887585>
23:30:35.702246 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 13033 win 997
        <nop,nop,timestamp 9336706 130887585>
23:30:36.306982 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 14481 win 1088
        <nop,nop,timestamp 9336857 130888189>
23:30:36.307016 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 34753 win 2003
        <nop,nop,timestamp 9336857 130889986>
23:30:36.911803 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 35953 win 2003
        <nop,nop,timestamp 9337009 130889987>
23:30:36.912241 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 37401 win 2003
        <nop,nop,timestamp 9337009 130890591>
23:30:37.466356 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 38849 win 2003
        <nop,nop,timestamp 9337147 130890591>
23:30:37.466390 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 40049 win 1965
        <nop,nop,timestamp 9337147 130890591>
23:30:38.021399 IP 10.100.100.7.36200 > 202.6.211.8.http: . ack 41003 win 2003
        <nop,nop,timestamp 9337286 130891047>
23:30:38.021957 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 15929 win 1178
        <nop,nop,timestamp 9337286 130891828>
23:30:38.526712 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 17377 win 1269
        <nop,nop,timestamp 9337412 130891828>
23:30:38.526748 IP 10.100.100.7.36203 > 202.6.211.8.http: . ack 17573 win 1359
        <nop,nop,timestamp 9337412 130892433>
23:30:38.552446 IP 10.100.100.7.36205 > 202.6.211.8.http: P 873:1268(395) ack 2767 win
        454 <nop,nop,timestamp 9337419 130885251>
23:30:39.456006 IP 10.100.100.7.36205 > 202.6.211.8.http: . ack 4197 win 545
        <nop,nop,timestamp 9337645 130895577>
23:30:39.458557 IP 10.100.100.7.36205 > 202.6.211.8.http: F 1268:1268(0) ack 4197 win
        545 <nop,nop,timestamp 9337645 130895577>
23:30:39.464134 IP 10.100.100.7.36205 > 202.6.211.8.http: R 3201167240:3201167240(0)
        win 0
23:30:39.464290 IP 10.100.100.7.36205 > 202.6.211.8.http: R 3201167241:3201167241(0)
        win 0
23:30:39.466131 IP 10.100.100.7.36204 > 202.6.211.8.http: P 877:1352(475) ack 2503 win
        342 <nop,nop,timestamp 9337647 130880047>
23:30:39.471527 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 3951 win 432
        <nop,nop,timestamp 9337648 130895595>
23:30:40.035013 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 5310 win 523
        <nop,nop,timestamp 9337789 130895595>
23:30:40.035167 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 5334 win 523
        <nop,nop,timestamp 9337789 130895603>
```

```
23:30:40.077408 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 6702 win 613
        <nop,nop,timestamp 9337800 130896198>
23:30:40.622988 IP 10.100.100.7.36204 > 202.6.211.8.http: . ack 6758 win 613
        <nop,nop,timestamp 9337936 130896205>
23:31:23.576918 IP 10.100.100.7.36202 > 202.6.211.8.http: . ack 10691 win 907
        <nop,nop,timestamp 9348675 130939683>

186 packets captured
372 packets received by filter
0 packets dropped by kernel
```

LAMPIRAN 2

Hasil tcpdump Dari *Server* www.klikbca.com Menuju Komputer *Client*

```
[root@Konoha ~]# tcpdump -ni eth2 ip dst 10.100.100.7 and ip src ! 172.18.3.194

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes

23:30:05.473683 IP 202.6.211.8.http > 10.100.100.7.36200: S 585278217:585278217(0) ack
        3187407279 win 5792 <mss 1460,sackOK,timestamp 130861582 9329149,nop,wscale
        2>
23:30:05.479147 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 400 win 1716
        <nop,nop,timestamp 130861586 9329150>
23:30:06.402907 IP 202.6.211.8.http > 10.100.100.7.36200: P 1:1390(1389) ack 400 win
        1716 <nop,nop,timestamp 130862507 9329150>
23:30:06.413439 IP 202.6.211.8.http > 10.100.100.7.36200: P 1390:2758(1368) ack 400
        win 1716 <nop,nop,timestamp 130862517 9329382>
23:30:06.464248 IP 202.6.211.8.http > 10.100.100.7.36201: S 587995200:587995200(0) ack
        3182956104 win 5792 <mss 1460,sackOK,timestamp 130862572 9329396,nop,wscale
        2>
23:30:06.467548 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 435 win 1716
        <nop,nop,timestamp 130862576 9329397>
23:30:06.471316 IP 202.6.211.8.http > 10.100.100.7.36201: . 1:1449(1448) ack 435 win
        1716 <nop,nop,timestamp 130862577 9329397>
23:30:06.987363 IP 202.6.211.8.http > 10.100.100.7.36201: . 1449:2897(1448) ack 435
        win 1716 <nop,nop,timestamp 130862577 9329397>
23:30:06.987393 IP 202.6.211.8.http > 10.100.100.7.36201: P 2897:3046(149) ack 435 win
        1716 <nop,nop,timestamp 130862582 9329399>
23:30:07.331235 IP 202.6.211.8.http > 10.100.100.7.36201: . 1449:2897(1448) ack 435
        win 1716 <nop,nop,timestamp 130862786 9329399>
23:30:07.331611 IP 202.6.211.8.http > 10.100.100.7.36201: . 3046:4494(1448) ack 856
        win 1984 <nop,nop,timestamp 130863107 9329529>
23:30:07.935914 IP 202.6.211.8.http > 10.100.100.7.36201: . 4494:5942(1448) ack 856
        win 1984 <nop,nop,timestamp 130863107 9329529>
23:30:07.936722 IP 202.6.211.8.http > 10.100.100.7.36201: P 5942:7094(1152) ack 856
        win 1984 <nop,nop,timestamp 130863107 9329529>
23:30:08.481334 IP 202.6.211.8.http > 10.100.100.7.36200: P 2758:4126(1368) ack 400
        win 1716 <nop,nop,timestamp 130863138 9329384>
23:30:08.481439 IP 202.6.211.8.http > 10.100.100.7.36200: P 4126:5494(1368) ack 400
        win 1716 <nop,nop,timestamp 130863146 9329384>
23:30:09.051837 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 856 win 1984
        <nop,nop,timestamp 130863307 9329580,nop,nop,sack 1 {435:856}>
23:30:09.053282 IP 202.6.211.8.http > 10.100.100.7.36200: P 2758:4126(1368) ack 400
        win 1716 <nop,nop,timestamp 130863342 9329384>
23:30:09.054559 IP 202.6.211.8.http > 10.100.100.7.36201: . 7094:8542(1448) ack 856
        win 1984 <nop,nop,timestamp 130863446 9329614>
23:30:09.655963 IP 202.6.211.8.http > 10.100.100.7.36201: . 8542:9990(1448) ack 856
        win 1984 <nop,nop,timestamp 130863446 9329614>
23:30:09.656921 IP 202.6.211.8.http > 10.100.100.7.36200: P 2758:4126(1368) ack 400
        win 1716 <nop,nop,timestamp 130863750 9329384>
23:30:10.244925 IP 202.6.211.8.http > 10.100.100.7.36201: . 9990:11438(1448) ack 856
        win 1984 <nop,nop,timestamp 130864051 9329765>
23:30:10.245127 IP 202.6.211.8.http > 10.100.100.7.36201: . 11438:12886(1448) ack 856
        win 1984 <nop,nop,timestamp 130864051 9329765>
23:30:10.849552 IP 202.6.211.8.http > 10.100.100.7.36201: . 12886:14334(1448) ack 856
        win 1984 <nop,nop,timestamp 130864051 9329765>
23:30:10.850102 IP 202.6.211.8.http > 10.100.100.7.36201: P 14334:15286(952) ack 856
        win 1984 <nop,nop,timestamp 130864051 9329765>
23:30:11.355306 IP 202.6.211.8.http > 10.100.100.7.36200: P 2758:4126(1368) ack 400
        win 1716 <nop,nop,timestamp 130864566 9329384>
23:30:11.355444 IP 202.6.211.8.http > 10.100.100.7.36200: P 4126:5494(1368) ack 400
        win 1716 <nop,nop,timestamp 130864594 9329901>
23:30:11.927065 IP 202.6.211.8.http > 10.100.100.7.36200: . 5494:6942(1448) ack 400
        win 1716 <nop,nop,timestamp 130864594 9329901>
23:30:11.927097 IP 202.6.211.8.http > 10.100.100.7.36200: P 6942:7137(195) ack 400 win
        1716 <nop,nop,timestamp 130864594 9329901>
23:30:12.283447 IP 202.6.211.8.http > 10.100.100.7.36201: . 15286:16734(1448) ack 856
        win 1984 <nop,nop,timestamp 130865167 9330044>
23:30:12.283600 IP 202.6.211.8.http > 10.100.100.7.36201: . 16734:18182(1448) ack 856
        win 1984 <nop,nop,timestamp 130865167 9330044>
```

```
23:30:12.885643 IP 202.6.211.8.http > 10.100.100.7.36201: . 18182:19630(1448) ack 856
        win 1984 <nop,nop,timestamp 130865770 9330195>
23:30:12.886843 IP 202.6.211.8.http > 10.100.100.7.36201: . 19630:21078(1448) ack 856
        win 1984 <nop,nop,timestamp 130865770 9330195>
23:30:13.490262 IP 202.6.211.8.http > 10.100.100.7.36201: . 21078:22526(1448) ack 856
        win 1984 <nop,nop,timestamp 130866358 9330342>
23:30:13.491029 IP 202.6.211.8.http > 10.100.100.7.36201: P 22526:23478(952) ack 856
        win 1984 <nop,nop,timestamp 130866358 9330342>
23:30:13.995022 IP 202.6.211.8.http > 10.100.100.7.36201: . 23478:24926(1448) ack 856
        win 1984 <nop,nop,timestamp 130866358 9330342>
23:30:13.996185 IP 202.6.211.8.http > 10.100.100.7.36201: . 24926:26374(1448) ack 856
        win 1984 <nop,nop,timestamp 130866358 9330342>
23:30:14.599861 IP 202.6.211.8.http > 10.100.100.7.36201: . 26374:27822(1448) ack 856
        win 1984 <nop,nop,timestamp 130866963 9330493>
23:30:14.599991 IP 202.6.211.8.http > 10.100.100.7.36201: . 27822:29270(1448) ack 856
        win 1984 <nop,nop,timestamp 130866963 9330493>
23:30:15.206086 IP 202.6.211.8.http > 10.100.100.7.36201: . 29270:30718(1448) ack 856
        win 1984 <nop,nop,timestamp 130866965 9330493>
23:30:15.206174 IP 202.6.211.8.http > 10.100.100.7.36201: P 30718:31670(952) ack 856
        win 1984 <nop,nop,timestamp 130866965 9330493>
23:30:15.709202 IP 202.6.211.8.http > 10.100.100.7.36200: . 5494:6942(1448) ack 400
        win 1716 <nop,nop,timestamp 130867298 9330195>
23:30:15.710360 IP 202.6.211.8.http > 10.100.100.7.36201: . 31670:33118(1448) ack 856
        win 1984 <nop,nop,timestamp 130868397 9330852>
23:30:16.314422 IP 202.6.211.8.http > 10.100.100.7.36201: P 33118:34133(1015) ack 856
        win 1984 <nop,nop,timestamp 130868397 9330852>
23:30:16.341636 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 820 win 1984
        <nop,nop,timestamp 130872454 9331865>
23:30:16.343308 IP 202.6.211.8.http > 10.100.100.7.36200: . 7137:8585(1448) ack 820
        win 1984 <nop,nop,timestamp 130872455 9331865>
23:30:16.844640 IP 202.6.211.8.http > 10.100.100.7.36200: . 8585:10033(1448) ack 820
        win 1984 <nop,nop,timestamp 130872455 9331865>
23:30:16.845501 IP 202.6.211.8.http > 10.100.100.7.36200: P 10033:11186(1153) ack 820
        win 1984 <nop,nop,timestamp 130872455 9331865>
23:30:17.391485 IP 202.6.211.8.http > 10.100.100.7.36200: . 11186:12634(1448) ack 820
        win 1984 <nop,nop,timestamp 130872459 9331867>
23:30:17.391599 IP 202.6.211.8.http > 10.100.100.7.36200: . 12634:14082(1448) ack 820
        win 1984 <nop,nop,timestamp 130872962 9331992>
23:30:17.996019 IP 202.6.211.8.http > 10.100.100.7.36200: P 14082:15429(1347) ack 820
        win 1984 <nop,nop,timestamp 130872962 9331992>
23:30:18.194206 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 1290 win 2252
        <nop,nop,timestamp 130874309 9332329>
23:30:18.196089 IP 202.6.211.8.http > 10.100.100.7.36202: S 597004164:597004164(0) ack
        3202392591 win 5792 <mss 1460,sackOK,timestamp 130874309 9332329,nop,wscale
        2>
23:30:18.196153 IP 202.6.211.8.http > 10.100.100.7.36203: S 591197557:591197557(0) ack
        3202746617 win 5792 <mss 1460,sackOK,timestamp 130874309 9332329,nop,wscale
        2>
23:30:18.199909 IP 202.6.211.8.http > 10.100.100.7.36201: . 34133:35581(1448) ack 1290
        win 2252 <nop,nop,timestamp 130874310 9332329>
23:30:18.619857 IP 202.6.211.8.http > 10.100.100.7.36201: . 35581:37029(1448) ack 1290
        win 2252 <nop,nop,timestamp 130874310 9332329>
23:30:18.619941 IP 202.6.211.8.http > 10.100.100.7.36201: P 37029:37917(888) ack 1290
        win 2252 <nop,nop,timestamp 130874310 9332329>
23:30:19.110817 IP 202.6.211.8.http > 10.100.100.7.36203: . ack 441 win 1716
        <nop,nop,timestamp 130874318 9332331>
23:30:19.110886 IP 202.6.211.8.http > 10.100.100.7.36202: . ack 438 win 1716
        <nop,nop,timestamp 130874318 9332331>
23:30:19.110985 IP 202.6.211.8.http > 10.100.100.7.36204: S 595130554:595130554(0) ack
        3208415530 win 5792 <mss 1460,sackOK,timestamp 130874318 9332331,nop,wscale
        2>
23:30:19.111014 IP 202.6.211.8.http > 10.100.100.7.36205: S 597494178:597494178(0) ack
        3201165972 win 5792 <mss 1460,sackOK,timestamp 130874318 9332331,nop,wscale
        2>
23:30:19.111068 IP 202.6.211.8.http > 10.100.100.7.36206: S 603847600:603847600(0) ack
        3205423462 win 5792 <mss 1460,sackOK,timestamp 130874318 9332331,nop,wscale
        2>
23:30:19.113567 IP 202.6.211.8.http > 10.100.100.7.36202: . 1:1449(1448) ack 438 win
        1716 <nop,nop,timestamp 130874319 9332331>
23:30:19.482713 IP 202.6.211.8.http > 10.100.100.7.36202: P 1449:2344(895) ack 438 win
        1716 <nop,nop,timestamp 130874319 9332331>
23:30:19.484502 IP 202.6.211.8.http > 10.100.100.7.36203: . 1:1449(1448) ack 441 win
        1716 <nop,nop,timestamp 130874319 9332331>
23:30:19.978194 IP 202.6.211.8.http > 10.100.100.7.36203: . 1449:2897(1448) ack 441
        win 1716 <nop,nop,timestamp 130874319 9332331>
```

```
23:30:19.978226 IP 202.6.211.8.http > 10.100.100.7.36207: S 595105992:595105992(0) ack
        3208163836 win 5792 <mss 1460,sackOK,timestamp 130874330 9332333,nop,wscale
        2>
23:30:19.978240 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 1254 win 2252
        <nop,nop,timestamp 130874349 9332329>
23:30:20.305228 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 1254 win 2252
        <nop,nop,timestamp 130874514 9332380,nop,nop,sack 1 {820:1254}>
23:30:20.305297 IP 202.6.211.8.http > 10.100.100.7.36203: . ack 441 win 1716
        <nop,nop,timestamp 130874521 9332382,nop,nop,sack 1 {1:441}>
23:30:20.305349 IP 202.6.211.8.http > 10.100.100.7.36202: . ack 438 win 1716
        <nop,nop,timestamp 130874522 9332382,nop,nop,sack 1 {1:438}>
23:30:20.307850 IP 202.6.211.8.http > 10.100.100.7.36202: . 1:1449(1448) ack 438 win
        1716 <nop,nop,timestamp 130874528 9332382>
23:30:20.653847 IP 202.6.211.8.http > 10.100.100.7.36203: . 1:1449(1448) ack 441 win
        1716 <nop,nop,timestamp 130874528 9332382>
23:30:20.653940 IP 202.6.211.8.http > 10.100.100.7.36201: P 37917:39035(1118) ack 1729
        win 2252 <nop,nop,timestamp 130874740 9332436>
23:30:21.191564 IP 202.6.211.8.http > 10.100.100.7.36200: . 15429:16877(1448) ack 1254
        win 2252 <nop,nop,timestamp 130874753 9332380>
23:30:21.192738 IP 202.6.211.8.http > 10.100.100.7.36200: . 16877:18325(1448) ack 1254
        win 2252 <nop,nop,timestamp 130874753 9332380>
23:30:21.796032 IP 202.6.211.8.http > 10.100.100.7.36200: P 18325:19569(1244) ack 1254
        win 2252 <nop,nop,timestamp 130874753 9332380>
23:30:21.797424 IP 202.6.211.8.http > 10.100.100.7.36200: . 19569:21017(1448) ack 1254
        win 2252 <nop,nop,timestamp 130874753 9332380>
23:30:22.357231 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 1254 win 2252
        <nop,nop,timestamp 130874921 9332482,nop,nop,sack 1 {820:1254}>
23:30:22.357320 IP 202.6.211.8.http > 10.100.100.7.36203: . ack 441 win 1716
        <nop,nop,timestamp 130874929 9332484,nop,nop,sack 1 {1:441}>
23:30:22.357372 IP 202.6.211.8.http > 10.100.100.7.36202: . ack 438 win 1716
        <nop,nop,timestamp 130874930 9332484,nop,nop,sack 1 {1:438}>
23:30:22.360090 IP 202.6.211.8.http > 10.100.100.7.36202: . 1:1449(1448) ack 438 win
        1716 <nop,nop,timestamp 130874944 9332484>
23:30:22.705603 IP 202.6.211.8.http > 10.100.100.7.36203: . 1:1449(1448) ack 441 win
        1716 <nop,nop,timestamp 130874944 9332484>
23:30:22.705632 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 1729 win 2252
        <nop,nop,timestamp 130875114 9332530,nop,nop,sack 1 {1290:1729}>
23:30:22.705647 IP 202.6.211.8.http > 10.100.100.7.36206: . ack 439 win 1716
        <nop,nop,timestamp 130875229 9332559>
23:30:23.031924 IP 202.6.211.8.http > 10.100.100.7.36205: . ack 439 win 1716
        <nop,nop,timestamp 130875229 9332559>
23:30:23.033762 IP 202.6.211.8.http > 10.100.100.7.36206: . 1:1449(1448) ack 439 win
        1716 <nop,nop,timestamp 130875230 9332559>
23:30:23.033787 IP 202.6.211.8.http > 10.100.100.7.36206: P 1449:1990(541) ack 439 win
        1716 <nop,nop,timestamp 130875230 9332559>
23:30:23.469608 IP 202.6.211.8.http > 10.100.100.7.36205: . 1:1449(1448) ack 439 win
        1716 <nop,nop,timestamp 130875230 9332559>
23:30:23.469646 IP 202.6.211.8.http > 10.100.100.7.36205: P 1449:1933(484) ack 439 win
        1716 <nop,nop,timestamp 130875230 9332559>
23:30:23.880916 IP 202.6.211.8.http > 10.100.100.7.36204: . ack 439 win 1716
        <nop,nop,timestamp 130875235 9332559>
23:30:23.882003 IP 202.6.211.8.http > 10.100.100.7.36202: P 1449:2344(895) ack 438 win
        1716 <nop,nop,timestamp 130875235 9332559>
23:30:23.883371 IP 202.6.211.8.http > 10.100.100.7.36204: P 1:1092(1091) ack 439 win
        1716 <nop,nop,timestamp 130875235 9332559>
23:30:24.313772 IP 202.6.211.8.http > 10.100.100.7.36202: . ack 878 win 1984
        <nop,nop,timestamp 130875601 9332652>
23:30:24.315114 IP 202.6.211.8.http > 10.100.100.7.36202: . 2344:3792(1448) ack 878
        win 1984 <nop,nop,timestamp 130875602 9332652>
23:30:24.316247 IP 202.6.211.8.http > 10.100.100.7.36202: . 3792:5240(1448) ack 878
        win 1984 <nop,nop,timestamp 130875602 9332652>
23:30:24.933843 IP 202.6.211.8.http > 10.100.100.7.36203: . 1449:2897(1448) ack 441
        win 1716 <nop,nop,timestamp 130875604 9332652>
23:30:24.934046 IP 202.6.211.8.http > 10.100.100.7.36203: . 2897:4345(1448) ack 441
        win 1716 <nop,nop,timestamp 130875604 9332652>
23:30:25.535027 IP 202.6.211.8.http > 10.100.100.7.36200: . ack 1254 win 2252
        <nop,nop,timestamp 130875737 9332686,nop,nop,sack 1 {820:1254}>
23:30:25.535075 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 1729 win 2252
        <nop,nop,timestamp 130875865 9332718,nop,nop,sack 1 {1290:1729}>
23:30:25.536536 IP 202.6.211.8.http > 10.100.100.7.36203: . 4345:5793(1448) ack 441
        win 1716 <nop,nop,timestamp 130876097 9332775>
23:30:25.868353 IP 202.6.211.8.http > 10.100.100.7.36203: P 5793:7241(1448) ack 441
        win 1716 <nop,nop,timestamp 130876097 9332775>
23:30:25.868380 IP 202.6.211.8.http > 10.100.100.7.36207: . ack 440 win 1716
        <nop,nop,timestamp 130876099 9332776>
23:30:25.869521 IP 202.6.211.8.http > 10.100.100.7.36207: . 1:1449(1448) ack 440 win
        1716 <nop,nop,timestamp 130876100 9332776>
```

```
23:30:26.484299 IP 202.6.211.8.http > 10.100.100.7.36207: P 1449:2228(779) ack 440 win
        1716 <nop,nop,timestamp 130876100 9332776>
23:30:26.484329 IP 202.6.211.8.http > 10.100.100.7.36202: . ack 878 win 1984
        <nop,nop,timestamp 130876631 9332909,nop,nop,sack 1 {438:878}>
23:30:26.484344 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 2165 win 2252
        <nop,nop,timestamp 130876773 9332944>
23:30:26.486213 IP 202.6.211.8.http > 10.100.100.7.36201: . 39035:40483(1448) ack 2165
        win 2252 <nop,nop,timestamp 130876774 9332944>
23:30:26.981457 IP 202.6.211.8.http > 10.100.100.7.36201: P 40483:40842(359) ack 2165
        win 2252 <nop,nop,timestamp 130876774 9332944>
23:30:26.983874 IP 202.6.211.8.http > 10.100.100.7.36200: . 21017:22465(1448) ack 1254
        win 2252 <nop,nop,timestamp 130877310 9333079>
23:30:27.369741 IP 202.6.211.8.http > 10.100.100.7.36200: . 22465:23913(1448) ack 1254
        win 2252 <nop,nop,timestamp 130877312 9333079>
23:30:27.369872 IP 202.6.211.8.http > 10.100.100.7.36202: . 2344:3792(1448) ack 878
        win 1984 <nop,nop,timestamp 130877719 9332909>
23:30:27.975386 IP 202.6.211.8.http > 10.100.100.7.36200: . 23913:25361(1448) ack 1254
        win 2252 <nop,nop,timestamp 130877916 9333230>
23:30:27.975558 IP 202.6.211.8.http > 10.100.100.7.36200: . 25361:26809(1448) ack 1254
        win 2252 <nop,nop,timestamp 130877917 9333230>
23:30:28.577257 IP 202.6.211.8.http > 10.100.100.7.36206: . ack 439 win 1716
        <nop,nop,timestamp 130877954 9333240,nop,nop,sack 1 {1:439}>
23:30:28.577366 IP 202.6.211.8.http > 10.100.100.7.36205: . ack 439 win 1716
        <nop,nop,timestamp 130877955 9333240,nop,nop,sack 1 {1:439}>
23:30:28.577418 IP 202.6.211.8.http > 10.100.100.7.36204: . ack 439 win 1716
        <nop,nop,timestamp 130877955 9333240,nop,nop,sack 1 {1:439}>
23:30:28.580111 IP 202.6.211.8.http > 10.100.100.7.36205: . 1:1449(1448) ack 439 win
        1716 <nop,nop,timestamp 130877960 9333240>
23:30:28.923919 IP 202.6.211.8.http > 10.100.100.7.36206: . 1:1449(1448) ack 439 win
        1716 <nop,nop,timestamp 130877963 9333240>
23:30:28.924628 IP 202.6.211.8.http > 10.100.100.7.36204: P 1:1092(1091) ack 439 win
        1716 <nop,nop,timestamp 130877965 9333240>
23:30:29.455095 IP 202.6.211.8.http > 10.100.100.7.36202: . ack 878 win 1984
        <nop,nop,timestamp 130878687 9333423,nop,nop,sack 1 {438:878}>
23:30:29.456650 IP 202.6.211.8.http > 10.100.100.7.36203: . 2897:4345(1448) ack 441
        win 1716 <nop,nop,timestamp 130878826 9333457>
23:30:29.456673 IP 202.6.211.8.http > 10.100.100.7.36206: . ack 874 win 1984
        <nop,nop,timestamp 130879155 9333539>
23:30:29.786258 IP 202.6.211.8.http > 10.100.100.7.36206: P 1990:3198(1208) ack 874
        win 1984 <nop,nop,timestamp 130879156 9333539>
23:30:29.786292 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 2165 win 2252
        <nop,nop,timestamp 130879280 9333571,nop,nop,sack 1 {1729:2165}>
23:30:29.786308 IP 202.6.211.8.http > 10.100.100.7.36205: . ack 874 win 1984
        <nop,nop,timestamp 130879591 9333648>
23:30:29.786844 IP 202.6.211.8.http > 10.100.100.7.36205: P 1933:2767(834) ack 874 win
        1984 <nop,nop,timestamp 130879592 9333648>
23:30:30.244436 IP 202.6.211.8.http > 10.100.100.7.36204: . ack 878 win 1984
        <nop,nop,timestamp 130880008 9333752>
23:30:30.246318 IP 202.6.211.8.http > 10.100.100.7.36204: P 1092:2503(1411) ack 878
        win 1984 <nop,nop,timestamp 130880047 9333752>
23:30:30.246521 IP 202.6.211.8.http > 10.100.100.7.36202: . 5240:6688(1448) ack 878
        win 1984 <nop,nop,timestamp 130880435 9333860>
23:30:30.857029 IP 202.6.211.8.http > 10.100.100.7.36202: . 6688:8136(1448) ack 878
        win 1984 <nop,nop,timestamp 130880435 9333860>
23:30:30.857151 IP 202.6.211.8.http > 10.100.100.7.36202: . 8136:9584(1448) ack 878
        win 1984 <nop,nop,timestamp 130880437 9333860>
23:30:31.461770 IP 202.6.211.8.http > 10.100.100.7.36203: . 7241:8689(1448) ack 441
        win 1716 <nop,nop,timestamp 130881054 9334014>
23:30:31.461898 IP 202.6.211.8.http > 10.100.100.7.36207: . 1:1449(1448) ack 440 win
        1716 <nop,nop,timestamp 130881401 9332776>
23:30:32.063608 IP 202.6.211.8.http > 10.100.100.7.36207: . ack 440 win 1716
        <nop,nop,timestamp 130881405 9334102,nop,nop,sack 1 {1:440}>
23:30:32.065420 IP 202.6.211.8.http > 10.100.100.7.36203: . 8689:10137(1448) ack 441
        win 1716 <nop,nop,timestamp 130881991 9334248>
23:30:32.065442 IP 202.6.211.8.http > 10.100.100.7.36207: . ack 873 win 1984
        <nop,nop,timestamp 130882606 9334402>
23:30:32.395139 IP 202.6.211.8.http > 10.100.100.7.36207: . 2228:3676(1448) ack 873
        win 1984 <nop,nop,timestamp 130882607 9334402>
23:30:32.395268 IP 202.6.211.8.http > 10.100.100.7.36207: . 3676:5124(1448) ack 873
        win 1984 <nop,nop,timestamp 130882607 9334402>
23:30:32.997549 IP 202.6.211.8.http > 10.100.100.7.36207: P 5124:5433(309) ack 873 win
        1984 <nop,nop,timestamp 130882607 9334402>
23:30:32.997571 IP 202.6.211.8.http > 10.100.100.7.36201: . ack 2598 win 2252
        <nop,nop,timestamp 130883104 9334526>
23:30:32.998949 IP 202.6.211.8.http > 10.100.100.7.36201: . 40842:42290(1448) ack 2598
        win 2252 <nop,nop,timestamp 130883105 9334526>
```

```
23:30:33.387783 IP 202.6.211.8.http > 10.100.100.7.36201: . 42290:43738(1448) ack 2598
        win 2252 <nop,nop,timestamp 130883105 9334526>
23:30:33.388147 IP 202.6.211.8.http > 10.100.100.7.36201: P 43738:44530(792) ack 2598
        win 2252 <nop,nop,timestamp 130883105 9334526>
23:30:33.860644 IP 202.6.211.8.http > 10.100.100.7.36200: . 26809:28257(1448) ack 1254
        win 2252 <nop,nop,timestamp 130883108 9334527>
23:30:33.860763 IP 202.6.211.8.http > 10.100.100.7.36200: . 28257:29705(1448) ack 1254
        win 2252 <nop,nop,timestamp 130883108 9334527>
23:30:34.465230 IP 202.6.211.8.http > 10.100.100.7.36200: . 29705:31153(1448) ack 1254
        win 2252 <nop,nop,timestamp 130883491 9334623>
23:30:34.465538 IP 202.6.211.8.http > 10.100.100.7.36200: P 31153:31857(704) ack 1254
        win 2252 <nop,nop,timestamp 130884097 9334775>
23:30:34.921039 IP 202.6.211.8.http > 10.100.100.7.36200: . 31857:33305(1448) ack 1254
        win 2252 <nop,nop,timestamp 130884097 9334775>
23:30:34.921066 IP 202.6.211.8.http > 10.100.100.7.36206: . ack 874 win 1984
        <nop,nop,timestamp 130884444 9334861,nop,nop,sack 1 {439:874}>
23:30:34.921081 IP 202.6.211.8.http > 10.100.100.7.36205: . ack 874 win 1984
        <nop,nop,timestamp 130885251 9335063,nop,nop,sack 1 {439:874}>
23:30:35.250217 IP 202.6.211.8.http > 10.100.100.7.36206: . ack 1315 win 2252
        <nop,nop,timestamp 130885912 9335228>
23:30:35.251256 IP 202.6.211.8.http > 10.100.100.7.36206: P 3198:4150(952) ack 1315
        win 2252 <nop,nop,timestamp 130885912 9335228>
23:30:35.252556 IP 202.6.211.8.http > 10.100.100.7.36202: P 9584:10690(1106) ack 878
        win 1984 <nop,nop,timestamp 130886370 9335342>
23:30:35.701760 IP 202.6.211.8.http > 10.100.100.7.36203: . 10137:11585(1448) ack 441
        win 1716 <nop,nop,timestamp 130887585 9335646>
23:30:35.701877 IP 202.6.211.8.http > 10.100.100.7.36203: . 11585:13033(1448) ack 441
        win 1716 <nop,nop,timestamp 130887585 9335646>
23:30:36.306473 IP 202.6.211.8.http > 10.100.100.7.36203: . 13033:14481(1448) ack 441
        win 1716 <nop,nop,timestamp 130888189 9335797>
23:30:36.306645 IP 202.6.211.8.http > 10.100.100.7.36200: . 33305:34753(1448) ack 1254
        win 2252 <nop,nop,timestamp 130889986 9336246>
23:30:36.911481 IP 202.6.211.8.http > 10.100.100.7.36200: P 34753:35953(1200) ack 1254
        win 2252 <nop,nop,timestamp 130889987 9336246>
23:30:36.911881 IP 202.6.211.8.http > 10.100.100.7.36200: . 35953:37401(1448) ack 1254
        win 2252 <nop,nop,timestamp 130890591 9336397>
23:30:37.465898 IP 202.6.211.8.http > 10.100.100.7.36200: . 37401:38849(1448) ack 1254
        win 2252 <nop,nop,timestamp 130890591 9336397>
23:30:37.465999 IP 202.6.211.8.http > 10.100.100.7.36200: P 38849:40049(1200) ack 1254
        win 2252 <nop,nop,timestamp 130890591 9336397>
23:30:38.021007 IP 202.6.211.8.http > 10.100.100.7.36200: P 40049:41003(954) ack 1254
        win 2252 <nop,nop,timestamp 130891047 9336511>
23:30:38.021585 IP 202.6.211.8.http > 10.100.100.7.36203: . 14481:15929(1448) ack 441
        win 1716 <nop,nop,timestamp 130891828 9336706>
23:30:38.526306 IP 202.6.211.8.http > 10.100.100.7.36203: . 15929:17377(1448) ack 441
        win 1716 <nop,nop,timestamp 130891828 9336706>
23:30:38.526332 IP 202.6.211.8.http > 10.100.100.7.36203: P 17377:17573(196) ack 441
        win 1716 <nop,nop,timestamp 130892433 9336857>
23:30:38.878442 IP 202.6.211.8.http > 10.100.100.7.36205: . ack 1269 win 2252
        <nop,nop,timestamp 130894681 9337419>
23:30:39.455558 IP 202.6.211.8.http > 10.100.100.7.36205: P 2767:4197(1430) ack 1269
        win 2252 <nop,nop,timestamp 130895577 9337419>
23:30:39.463648 IP 202.6.211.8.http > 10.100.100.7.36205: P 4197:5565(1368) ack 1269
        win 2252 <nop,nop,timestamp 130895585 9337645>
23:30:39.464203 IP 202.6.211.8.http > 10.100.100.7.36205: F 5565:5565(0) ack 1270 win
        2252 <nop,nop,timestamp 130895589 9337645>
23:30:39.471017 IP 202.6.211.8.http > 10.100.100.7.36204: . 2503:3951(1448) ack 1353
        win 2252 <nop,nop,timestamp 130895595 9337647>
23:30:40.034536 IP 202.6.211.8.http > 10.100.100.7.36204: P 3951:5310(1359) ack 1353
        win 2252 <nop,nop,timestamp 130895595 9337647>
23:30:40.034572 IP 202.6.211.8.http > 10.100.100.7.36204: P 5310:5334(24) ack 1353 win
        2252 <nop,nop,timestamp 130895603 9337648>
23:30:40.076924 IP 202.6.211.8.http > 10.100.100.7.36204: P 5334:6702(1368) ack 1353
        win 2252 <nop,nop,timestamp 130896198 9337789>
23:30:40.622651 IP 202.6.211.8.http > 10.100.100.7.36204: P 6702:6758(56) ack 1353 win
        2252 <nop,nop,timestamp 130896205 9337789>

174 packets captured
348 packets received by filter
0 packets dropped by kernel
```

LAMPIRAN 3

Hasil tcpdump Dari Komputer *Client* Menuju *Server* www.bni.co.id

```
[root@Konoha squid]# tcpdump -nni eth2 ip src 10.100.100.7 and dst port 80

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes

08:08:28.300994 IP 10.100.100.7.44838 > 219.83.38.10.80: S 829824112:829824112(0) win
        5840 <mss 1460,sackOK,timestamp 10804818 0,nop,wscale 5>
08:08:28.301088 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack 1920127707 win 183
        <nop,nop,timestamp 10804818 15461601>
08:08:28.301724 IP 10.100.100.7.44838 > 219.83.38.10.80: P 0:397(397) ack 1 win 183
        <nop,nop,timestamp 10804818 15461601>
08:08:28.315935 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack 1449 win 273
        <nop,nop,timestamp 10804822 15461604>
08:08:28.315998 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack 2447 win 364
        <nop,nop,timestamp 10804822 15461604>
08:08:28.483376 IP 10.100.100.7.44839 > 219.83.38.10.80: S 832390573:832390573(0) win
        5840 <mss 1460,sackOK,timestamp 10804864 0,nop,wscale 5>
08:08:28.483606 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 1919844599 win 183
        <nop,nop,timestamp 10804864 15461646>
08:08:28.483946 IP 10.100.100.7.44839 > 219.83.38.10.80: P 0:358(358) ack 1 win 183
        <nop,nop,timestamp 10804864 15461646>
08:08:28.496202 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 1449 win 273
        <nop,nop,timestamp 10804867 15461649>
08:08:28.496287 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 1814 win 364
        <nop,nop,timestamp 10804867 15461649>
08:08:28.888750 IP 10.100.100.7.44839 > 219.83.38.10.80: P 358:729(371) ack 1814 win
        364 <nop,nop,timestamp 10804965 15461649>
08:08:28.895609 IP 10.100.100.7.44838 > 219.83.38.10.80: P 397:725(328) ack 2447 win
        364 <nop,nop,timestamp 10804967 15461604>
08:08:28.896547 IP 10.100.100.7.44840 > 219.83.38.10.80: S 836860594:836860594(0) win
        5840 <mss 1460,sackOK,timestamp 10804967 0,nop,wscale 5>
08:08:28.896711 IP 10.100.100.7.44841 > 219.83.38.10.80: S 840051435:840051435(0) win
        5840 <mss 1460,sackOK,timestamp 10804967 0,nop,wscale 5>
08:08:28.896828 IP 10.100.100.7.44840 > 219.83.38.10.80: . ack 1922269594 win 183
        <nop,nop,timestamp 10804967 15461750>
08:08:28.896898 IP 10.100.100.7.44841 > 219.83.38.10.80: . ack 1915351374 win 183
        <nop,nop,timestamp 10804967 15461750>
08:08:28.897021 IP 10.100.100.7.44842 > 219.83.38.10.80: S 845376640:845376640(0) win
        5840 <mss 1460,sackOK,timestamp 10804967 0,nop,wscale 5>
08:08:28.897378 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 3262 win 454
        <nop,nop,timestamp 10804967 15461750>
08:08:28.897432 IP 10.100.100.7.44842 > 219.83.38.10.80: . ack 1917060627 win 183
        <nop,nop,timestamp 10804967 15461750>
08:08:28.897472 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 3415 win 545
        <nop,nop,timestamp 10804967 15461750>
08:08:28.897536 IP 10.100.100.7.44841 > 219.83.38.10.80: P 0:378(378) ack 1 win 183
        <nop,nop,timestamp 10804967 15461750>
08:08:28.897652 IP 10.100.100.7.44840 > 219.83.38.10.80: P 0:373(373) ack 1 win 183
        <nop,nop,timestamp 10804967 15461750>
08:08:28.899684 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack 2479 win 364
        <nop,nop,timestamp 10804968 15461750>
08:08:28.899932 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 4863 win 635
        <nop,nop,timestamp 10804968 15461750>
08:08:28.938425 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack 2601 win 364
        <nop,nop,timestamp 10804978 15461750>
08:08:29.010947 IP 10.100.100.7.44842 > 219.83.38.10.80: P 0:380(380) ack 1 win 183
        <nop,nop,timestamp 10804996 15461750>
08:08:29.011461 IP 10.100.100.7.44838 > 219.83.38.10.80: F 725:725(0) ack 2601 win 364
        <nop,nop,timestamp 10804996 15461750>
08:08:29.011797 IP 10.100.100.7.44838 > 219.83.38.10.80: . ack 2602 win 364
        <nop,nop,timestamp 10804996 15461778>
08:08:29.459004 IP 10.100.100.7.44839 > 219.83.38.10.80: . ack 5859 win 726
        <nop,nop,timestamp 10805108 15461890>
08:08:29.471415 IP 10.100.100.7.44840 > 219.83.38.10.80: . ack 1449 win 273
        <nop,nop,timestamp 10805111 15461893>
08:08:29.471478 IP 10.100.100.7.44840 > 219.83.38.10.80: . ack 1602 win 364
        <nop,nop,timestamp 10805111 15461893>
08:08:30.054577 IP 10.100.100.7.44840 > 219.83.38.10.80: . ack 3050 win 454
        <nop,nop,timestamp 10805257 15462039>
```

```
08:08:30.054649 IP 10.100.100.7.44840 > 219.83.38.10.80: . ack 3666 win 545
        <nop,nop,timestamp 10805257 15462039>
08:08:30.491954 IP 10.100.100.7.44841 > 219.83.38.10.80: . ack 783 win 232
        <nop,nop,timestamp 10805366 15462148>
08:08:30.493023 IP 10.100.100.7.44842 > 219.83.38.10.80: . ack 1449 win 273
        <nop,nop,timestamp 10805366 15462149>
08:08:30.493105 IP 10.100.100.7.44842 > 219.83.38.10.80: . ack 1602 win 364
        <nop,nop,timestamp 10805366 15462149>
08:08:30.930840 IP 10.100.100.7.44842 > 219.83.38.10.80: . ack 2079 win 454
        <nop,nop,timestamp 10805476 15462258>

37 packets captured
74 packets received by filter
0 packets dropped by kernel
```

LAMPIRAN 4

Hasil tcpdump Dari *Server* www.bni.co.id Menuju Komputer *Client*

```
[root@Konoha squid]# tcpdump -nni eth2 ip dst 10.100.100.7 and ip src ! 172.18.3.194

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes

01:08:28.301386 IP 219.83.38.10.80 > 10.100.100.7.44838: S 1920127706:1920127706(0)
        ack 829824113 win 5792 <mss 1460,sackOK,timestamp 15461601
        10804818,nop,wscale 5>
01:08:28.301824 IP 219.83.38.10.80 > 10.100.100.7.44838: . ack 398 win 215
        <nop,nop,timestamp 15461601 10804818>
01:08:28.314455 IP 219.83.38.10.80 > 10.100.100.7.44838: . 1:1449(1448) ack 398 win
        215 <nop,nop,timestamp 15461604 10804818>
01:08:28.314504 IP 219.83.38.10.80 > 10.100.100.7.44838: P 1449:2447(998) ack 398 win
        215 <nop,nop,timestamp 15461604 10804818>
01:08:28.483457 IP 219.83.38.10.80 > 10.100.100.7.44839: S 1919844598:1919844598(0)
        ack 832390574 win 5792 <mss 1460,sackOK,timestamp 15461646
        10804864,nop,wscale 5>
01:08:28.483990 IP 219.83.38.10.80 > 10.100.100.7.44839: . ack 359 win 215
        <nop,nop,timestamp 15461647 10804864>
01:08:28.495546 IP 219.83.38.10.80 > 10.100.100.7.44839: . 1:1449(1448) ack 359 win
        215 <nop,nop,timestamp 15461649 10804864>
01:08:28.495591 IP 219.83.38.10.80 > 10.100.100.7.44839: P 1449:1814(365) ack 359 win
        215 <nop,nop,timestamp 15461649 10804864>
01:08:28.896672 IP 219.83.38.10.80 > 10.100.100.7.44840: S 1922269593:1922269593(0)
        ack 836860595 win 5792 <mss 1460,sackOK,timestamp 15461750
        10804967,nop,wscale 5>
01:08:28.896742 IP 219.83.38.10.80 > 10.100.100.7.44841: S 1915351373:1915351373(0)
        ack 840051436 win 5792 <mss 1460,sackOK,timestamp 15461750
        10804967,nop,wscale 5>
01:08:28.896976 IP 219.83.38.10.80 > 10.100.100.7.44839: . 1814:3262(1448) ack 730 win
        248 <nop,nop,timestamp 15461750 10804965>
01:08:28.897066 IP 219.83.38.10.80 > 10.100.100.7.44842: S 1917060626:1917060626(0)
        ack 845376641 win 5792 <mss 1460,sackOK,timestamp 15461750
        10804967,nop,wscale 5>
01:08:28.897087 IP 219.83.38.10.80 > 10.100.100.7.44839: P 3262:3415(153) ack 730 win
        248 <nop,nop,timestamp 15461750 10804965>
01:08:28.897576 IP 219.83.38.10.80 > 10.100.100.7.44841: . ack 379 win 215
        <nop,nop,timestamp 15461750 10804967>
01:08:28.897683 IP 219.83.38.10.80 > 10.100.100.7.44840: . ack 374 win 215
        <nop,nop,timestamp 15461750 10804967>
01:08:28.899403 IP 219.83.38.10.80 > 10.100.100.7.44838: P 2447:2479(32) ack 726 win
        248 <nop,nop,timestamp 15461750 10804967>
01:08:28.899548 IP 219.83.38.10.80 > 10.100.100.7.44839: P 3415:4863(1448) ack 730 win
        248 <nop,nop,timestamp 15461750 10804967>
01:08:28.899604 IP 219.83.38.10.80 > 10.100.100.7.44838: P 2479:2601(122) ack 726 win
        248 <nop,nop,timestamp 15461750 10804967>
01:08:29.011012 IP 219.83.38.10.80 > 10.100.100.7.44842: . ack 381 win 215
        <nop,nop,timestamp 15461778 10804996>
01:08:29.011673 IP 219.83.38.10.80 > 10.100.100.7.44838: F 2601:2601(0) ack 727 win
        248 <nop,nop,timestamp 15461778 10804996>
01:08:29.012485 IP 172.17.63.129.8080 > 10.100.100.7.41635: S 1690024354:1690024354(0)
        ack 832540764 win 65535 <mss 1460,nop,wscale 1,nop,nop,timestamp 137020676
        10804996,sackOK,eol>
01:08:29.015242 IP 172.17.63.129.8080 > 10.100.100.7.41635: P 1:1347(1346) ack 403 win
        33304 <nop,nop,timestamp 137020679 10804996>
01:08:29.458506 IP 219.83.38.10.80 > 10.100.100.7.44839: P 4863:5859(996) ack 730 win
        248 <nop,nop,timestamp 15461890 10804968>
01:08:29.470908 IP 219.83.38.10.80 > 10.100.100.7.44840: . 1:1449(1448) ack 374 win
        215 <nop,nop,timestamp 15461893 10804967>
01:08:29.470950 IP 219.83.38.10.80 > 10.100.100.7.44840: P 1449:1602(153) ack 374 win
        215 <nop,nop,timestamp 15461893 10804967>
01:08:30.054150 IP 219.83.38.10.80 > 10.100.100.7.44840: P 1602:3050(1448) ack 374 win
        215 <nop,nop,timestamp 15462039 10805111>
01:08:30.054379 IP 219.83.38.10.80 > 10.100.100.7.44840: P 3050:3666(616) ack 374 win
        215 <nop,nop,timestamp 15462039 10805111>
01:08:30.112407 IP 172.17.63.129.8080 > 10.100.100.7.41635: P 1347:2692(1345) ack 805
        win 33304 <nop,nop,timestamp 137021776 10805271>
01:08:30.491521 IP 219.83.38.10.80 > 10.100.100.7.44841: P 1:783(782) ack 379 win 215
        <nop,nop,timestamp 15462148 10804967>
```

```
01:08:30.492574 IP 219.83.38.10.80 > 10.100.100.7.44842: . 1:1449(1448) ack 381 win
        215 <nop,nop,timestamp 15462149 10804996>
01:08:30.492612 IP 219.83.38.10.80 > 10.100.100.7.44842: P 1449:1602(153) ack 381 win
        215 <nop,nop,timestamp 15462149 10804996>
01:08:30.930541 IP 219.83.38.10.80 > 10.100.100.7.44842: P 1602:2079(477) ack 381 win
        215 <nop,nop,timestamp 15462258 10805366>

32 packets captured
64 packets received by filter
0 packets dropped by kernel
```

LAMPIRAN 5

Isi dari *file* /var/log/squid/access.log pada saat mengakses

www.bangbrosnetwork.com

```
[root@Konoha squid]# tail -f access.log

1190262549.123   1413 10.100.100.7 TCP_MISS/200 8463 GET
                 http://www.bangbrosnetwork.com/ - DIRECT/66.230.129.242 text/html
1190262549.729   288 10.100.100.7 TCP_MISS/200 2272 GET
                 http://xml.alexa.com/data? -DIRECT/64.213.200.101 text/xml
1190262551.254   50 10.100.100.7 TCP_HIT/200 1723 GET
                 http://images1.bangbros.com/shared/icra.gif - NONE/- image/gif
1190262552.830   2928 10.100.100.7 TCP_MISS/200 24408 GET
                 http://images2.bangbros.com/bangbrosnetwork/t1/2.jpg -
                 DIRECT/66.230.128.210 image/jpeg
1190262552.851   2949 10.100.100.7 TCP_MISS/200 32598 GET
                 http://images2.bangbros.com/bangbrosnetwork/t1/1.jpg -
                 DIRECT/66.230.128.210
                 image/jpeg
1190262554.742   591 10.100.100.7 TCP_MISS/200 284 GET
                 http://www.bangbrosnetwork.com/favicon.ico - DIRECT/66.230.188.2
                 image/x-icon
1190262554.974   5533 10.100.100.7 TCP_HIT/200 23666 GET
                 http://images1.bangbros.com/bangbus/t2/back2.jpg - NONE/- image/jpeg
1190262821.638   524 10.100.100.7 TCP_MISS/200 3100 GET
                 http://sb.google.com/safebrowsing/update? - DIRECT/209.85.163.91
                 text/html
1190263835.120   840 10.100.100.7 TCP_MISS/200 6000 GET
                 http://sb.google.com/safebrowsing/update? - DIRECT/209.85.163.91
                 text/html
1190264392.050   236 10.100.100.7 TCP_MISS/200 2347 GET
                 http://xml.alexa.com/data? - DIRECT/64.213.200.101 text/xml
1190264398.328   1305 10.100.100.7 TCP_MISS/200 8351 GET
                 http://www.bangbrosnetwork.com/ - DIRECT/66.230.188.6 text/html
1190264398.797   363 10.100.100.7 TCP_MISS/200 2273 GET
                 http://xml.alexa.com/data? - DIRECT/64.213.200.101 text/xml
1190264399.088   3 10.100.100.7 TCP_HIT/200 1723 GET
                 http://images1.bangbros.com/shared/icra.gif - NONE/- image/gif
1190264407.537   8697 10.100.100.7 TCP_HIT/200 52102 GET
                 http://images2.bangbros.com/bangbrosnetwork/t1/1.jpg - NONE/-
                 image/jpeg
1190264409.528   10687 10.100.100.7 TCP_HIT/200 57820 GET
                 http://images2.bangbros.com/bangbrosnetwork/t1/2.jpg - NONE/-
                 image/jpeg
1190264611.969   529 10.100.100.7 TCP_MISS/200 213 GET
                 http://sb.google.com/safebrowsing/update? - DIRECT/209.85.163.91
                 text/html
```

LAMPIRAN 6

Isi dari *file* /var/log/squid/access.log pada saat mengakses

www.bintangmawar.net/forum

```
[root@Konoha squid]# tailf access.log

1190347148.495   1152 10.100.100.7 TCP_MISS/200 4078 GET
                http://www.bintangmawar.net/favicon.ico - DIRECT/63.243.152.34
                image/x-icon
1190347162.976  21703 10.100.100.7 TCP_MISS/200 8517 GET http://www.bintangmawar.net/
                - DIRECT/63.243.152.34 text/html
1190347181.689  18891 10.100.100.7 TCP_MISS/302 696 GET
                http://ads.bintangmawar.net/adx.js - DIRECT/72.249.47.176 text/html
1190347182.000    285 10.100.100.7 TCP_MISS/404 682 GET
                http://ns5.fastnet5.com/suspended.page/ - DIRECT/65.99.213.7 text/html
1190347186.224   4213 10.100.100.7 TCP_MISS/302 772 GET
                http://ads.bintangmawar.net/adjs.php? - DIRECT/72.249.47.176 text/html
1190347187.182    940 10.100.100.7 TCP_MISS/404 673 GET
                http://ns5.fastnet5.com/suspended.page/? - DIRECT/65.99.213.7
                text/html
1190347281.670    289 10.100.100.7 TCP_MISS/200 4091 GET
                http://www.bintangmawar.net/forum/favicon.ico - DIRECT/63.243.152.34
                image/x-icon
1190347286.077  14239 10.100.100.7 TCP_MISS/200 19231 GET
                http://www.bintangmawar.net/forum/come_inside.php -
                DIRECT/63.243.152.34 text/html
1190347295.412  12351 10.100.100.7 TCP_MISS/200 44352 GET
                http://www.bintangmawar.net/forum/clientscript/vbulletin_global.js? -
                DIRECT/63.243.152.34 application/x-javascript
1190347306.402  10988 10.100.100.7 TCP_MISS/200 18305 GET
                http://www.bintangmawar.net/forum/clientscript/vbulletin_menu.js? -
                DIRECT/63.243.152.34 application/x-javascript
1190347306.883    459 10.100.100.7 TCP_MISS/200 554 GET
                http://www.bintangmawar.net/forum/close.gif - DIRECT/63.243.152.34
                image/gif
1190347311.277   4831 10.100.100.7 TCP_MISS/200 20736 GET
                http://www.bintangmawar.net/forum/images_black/misc/vbulletin3_logo_bl
                ack.gif - DIRECT/63.243.152.34 image/gif
1190347312.259    981 10.100.100.7 TCP_MISS/200 1454 GET
                http://www.bintangmawar.net/forum/images_black/misc/navbits_start.gif
                - DIRECT/63.243.152.34 image/gif
1190347319.262  12379 10.100.100.7 TCP_MISS/200 10115 GET
                http://www.bintangmawar.net/forum/clientscript/vbulletin_md5.js? -
                DIRECT/63.243.152.34 application/x-javascript
1190347319.477    151 10.100.100.7 TCP_MISS/200 539 GET
                http://www.bintangmawar.net/forum/images_black/buttons/collapse_tcat.g
                if - DIRECT/63.243.152.34 image/gif
1190347331.941  12464 10.100.100.7 TCP_MISS/200 7245 GET
                http://www.bintangmawar.net/forum/clientscript/vbulletin_read_marker.j
                s? - DIRECT/63.243.152.34 application/x-javascript
1190347334.326  15063 10.100.100.7 TCP_MISS/200 63591 GET
                http://www.bintangmawar.net/forum/bannerimages/dermaga/dermaga750x70.g
                if - DIRECT/63.243.152.34 image/gif
1190347334.695   2754 10.100.100.7 TCP_MISS/200 2094 GET
                http://www.bintangmawar.net/forum/images_black/statusicon/forum_old.gi
                f - DIRECT/63.243.152.34 image/gif
1190347334.899    204 10.100.100.7 TCP_MISS/200 1482 GET
                http://www.bintangmawar.net/forum/images/icons/icon1.gif -
                DIRECT/63.243.152.34 image/gif
1190347335.138    239 10.100.100.7 TCP_MISS/200 1016 GET
                http://www.bintangmawar.net/forum/images_black/buttons/lastpost.gif -
                DIRECT/63.243.152.34 image/gif
1190347335.273    947 10.100.100.7 TCP_MISS/200 1111 GET
                http://www.bintangmawar.net/forum/bannerimages/updates-logo.gif -
                DIRECT/63.243.152.34 image/gif
1190347335.307     34 10.100.100.7 TCP_MISS/200 1123 GET
                http://www.bintangmawar.net/updates.gif - DIRECT/63.243.152.34
                image/gif
1190347335.730    422 10.100.100.7 TCP_MISS/200 1507 GET
                http://www.bintangmawar.net/forum/images/icons/icon5.gif -
                DIRECT/63.243.152.34 image/gif
```

```
1190347335.892    161 10.100.100.7 TCP_MISS/200 1473 GET
                  http://www.bintangmawar.net/forum/images/icons/icon14.gif -
                  DIRECT/63.243.152.34 image/gif
1190347337.539   1647 10.100.100.7 TCP_MISS/200 6719 GET
                  http://www.bintangmawar.net/forum/bannerimages/sdsb/bm_sdsb150x40.gif
                  - DIRECT/63.243.152.34 image/gif
1190347337.541   2402 10.100.100.7 TCP_MISS/200 1469 GET
                  http://www.bintangmawar.net/forum/images/icons/icon4.gif -
                  DIRECT/63.243.152.34 image/gif
1190347343.506   5967 10.100.100.7 TCP_MISS/200 16825 GET
                  http://www.bintangmawar.net/forum/bannerimages/4dtogel/banner_4dtogel.
                  gif - DIRECT/63.243.152.34 image/gif
1190347347.012   3506 10.100.100.7 TCP_MISS/200 5195 GET
                  http://www.bintangmawar.net/forum/bannerimages/bm-rentspace-90k.gif -
                  DIRECT/63.243.152.34 image/gif
1190347349.190  11649 10.100.100.7 TCP_MISS/200 32948 GET
                  http://www.bintangmawar.net/forum/bannerimages/dermaga/dermaga150x40.g
                  if - DIRECT/63.243.152.34 image/gif
1190347351.283   4270 10.100.100.7 TCP_MISS/200 8843 GET
                  http://www.bintangmawar.net/forum/bannerimages/bm_rentv2-90k.gif -
                  DIRECT/63.243.152.34 image/gif
1190347353.198   1915 10.100.100.7 TCP_MISS/200 1498 GET
                  http://www.bintangmawar.net/forum/images/icons/icon12.gif -
                  DIRECT/63.243.152.34 image/gif
1190347359.731  10539 10.100.100.7 TCP_MISS/200 25052 GET
                  http://www.bintangmawar.net/forum/bannerimages/668online/668online-
                  150px.gif - DIRECT/63.243.152.34 image/gif
1190347369.162   9432 10.100.100.7 TCP_MISS/200 532 GET
                  http://www.bintangmawar.net/forum/images_black/buttons/collapse_thead.
                  gif - DIRECT/63.243.152.34 image/gif
1190347381.671  28472 10.100.100.7 TCP_MISS/200 119041 GET
                  http://www.bintangmawar.net/forum/bannerimages/banner-profmbr2.gif -
                  DIRECT/63.243.152.34 image/gif
1190347382.260    589 10.100.100.7 TCP_MISS/200 1911 GET
                  http://www.bintangmawar.net/forum/images_black/misc/stats.gif -
                  DIRECT/63.243.152.34 image/gif
1190347382.652    390 10.100.100.7 TCP_MISS/200 2088 GET
                  http://www.bintangmawar.net/forum/images_black/statusicon/forum_new.gi
                  f - DIRECT/63.243.152.34 image/gif
1190347383.367    715 10.100.100.7 TCP_MISS/200 3235 GET
                  http://www.bintangmawar.net/forum/images_black/gradients/gradient_tcat
                  .gif - DIRECT/63.243.152.34 image/gif
1190347383.499    131 10.100.100.7 TCP_MISS/200 2494 GET
                  http://www.bintangmawar.net/forum/images_black/gradients/gradient_thea
                  d.gif - DIRECT/63.243.152.34 image/gif
1190347396.105  26942 10.100.100.7 TCP_MISS/200 1890 GET
                  http://www.bintangmawar.net/forum/images_black/misc/whos_online.gif -
                  DIRECT/63.243.152.34 image/gif
```

LAMPIRAN 7

Isi dari *file* squid.conf

```
http_port 3128 transparent
ssl_unclean_shutdown off
icp_port 3130
htcp_port 4827
udp_incoming_address 0.0.0.0
udp_outgoing_address 255.255.255.255
icp_query_timeout 0
maximum_icp_query_timeout 2000
mcast_icp_query_timeout 2000
dead_peer_timeout 10 seconds
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
cache_vary on
acl apache rep_header Server ^Apache
broken_vary_encoding allow apache
cache_mem 32 MB
cache_swap_low 90
cache_swap_high 95
minimum_object_size 102400 KB
maximum_object_size 102401 KB
maximum_object_size_in_memory 8 KB
ipcache_size 1024
ipcache_low 90
ipcache_high 95
fqdncache_size 1024
cache_replacement_policy lru
memory_replacement_policy lru
cache_dir ufs /var/spool/squid 50 16 256
access_log /var/log/squid/access.log squid
cache_log /var/log/squid/cache.log
cache_store_log /var/log/squid/store.log
emulate_httpd_log off
log_ip_on_direct on
mime_table /etc/squid/mime.conf
log_mime_hdrs off
pid_filename /var/run/squid.pid
debug_options ALL,1
log_fqdn off
client_netmask 255.255.255.255
ftp_user Squid@
ftp_list_width 32
ftp_passive on
ftp_sanitycheck on
ftp_telnet_protocol on
check_hostnames on
allow_underscore on
cache_dns_program /usr/lib/squid/dnsserver
dns_children 5
dns_retransmit_interval 5 seconds
dns_timeout 2 minutes
dns_defnames off
hosts_file /etc/hosts
diskd_program /usr/lib/squid/diskd-daemon
unlinkd_program /usr/lib/squid/unlinkd
pinger_program /usr/lib/squid/pinger
url_rewrite_program /usr/bin/squidGuard -c /etc/squid/squidGuard.conf
url_rewrite_children 25
url_rewrite_concurrency 0
url_rewrite_host_header on
location_rewrite_children 5
location_rewrite_concurrency 0
authenticate_ttl 1 hour
authenticate_ip_ttl 0 seconds
wais_relay_port 0
request_header_max_size 20 KB
request_body_max_size 0 KB
refresh_pattern ^ftp:  1440    20%     10080
refresh_pattern ^gopher:       1440    0%      1440
refresh_pattern .      0       20%     4320
```

```
quick_abort_min 16 KB
quick_abort_max 16 KB
quick_abort_pct 95
read_ahead_gap 16 KB
negative_ttl 5 minutes
positive_dns_ttl 6 hours
negative_dns_ttl 1 minute
range_offset_limit 0 KB
collapsed_forwarding off
refresh_stale_hit 0 seconds
forward_timeout 4 minutes
connect_timeout 1 minute
peer_connect_timeout 30 seconds
read_timeout 15 minutes
request_timeout 5 minutes
persistent_request_timeout 1 minute
client_lifetime 1 day
half_closed_clients on
pconn_timeout 120 seconds
ident_timeout 10 seconds
shutdown_lifetime 30 seconds
acl all src 0.0.0.0/0.0.0.0
acl net1 src 172.17.63.128/25
acl net2 src 10.100.100.0/25
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443 563     # https, snews
acl Safe_ports port 70          # gopher
acl Safe_ports port 210                 # wais
acl Safe_ports port 1025-65535          # unregistered ports
acl Safe_ports port 280                 # http-mgmt
acl Safe_ports port 488                 # gss-http
acl Safe_ports port 591                 # filemaker
acl Safe_ports port 777                 # multiling http
acl CONNECT method CONNECT
follow_x_forwarded_for deny all
acl_uses_indirect_client on
delay_pool_uses_indirect_client on
log_uses_indirect_client on
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access allow net1
http_access allow net2
http_access deny all
http_reply_access allow all
icp_access allow all
htcp_access deny all
htcp_clr_access deny all
miss_access allow all
ident_lookup_access deny all
visible_hostname Konohagakure-no-Proxy
reply_header_max_size 20 KB
reply_body_max_size 0 allow all
cache_mgr root
mail_program mail
cache_effective_user squid
cache_effective_group squid
httpd_suppress_version_string off
umask 027
announce_period 0
announce_period 1 day
announce_host tracker.ircache.net
announce_port 3131
httpd_accel_no_pmtu_disc off
dns_testnames netscape.com internic.net nlanr.net microsoft.com
logfile_rotate 0
append_domain .yourdomain.com
tcp_recv_bufsize 0 bytes
memory_pools on
memory_pools_limit 5 MB
```

```
via on
forwarded_for on
log_icp_queries on
icp_hit_stale off
minimum_direct_hops 4
minimum_direct_rtt 400
store_avg_object_size 13 KB
store_objects_per_bucket 20
client_db on
netdb_low 900
netdb_high 1000
netdb_ping_period 5 minutes
query_icmp off
test_reachability off
buffered_logs off
reload_into_ims off
icon_directory /usr/share/squid/icons
global_internal_static on
short_icon_urls off
error_directory /usr/share/squid/errors/English
maximum_single_addr_tries 1
retry_on_error off
snmp_port 0
snmp_access deny all
snmp_incoming_address 0.0.0.0
snmp_outgoing_address 255.255.255.255
as_whois_server whois.ra.net
as_whois_server whois.ra.net
wccp_router 0.0.0.0
wccp_version 4
wccp2_rebuild_wait on
wccp2_forwarding_method 1
wccp2_return_method 1
wccp2_assignment_method 1
wccp2_service standard 0
wccp2_weight 10000
wccp_address 0.0.0.0
wccp2_address 0.0.0.0
delay_pools 0
delay_initial_bucket_level 50
max_open_disk_fds 0
offline_mode off
uri_whitespace strip
mcast_miss_addr 255.255.255.255
mcast_miss_ttl 16
mcast_miss_port 3135
mcast_miss_encode_key XXXXXXXXXXXXXXXX
nonhierarchical_direct on
prefer_direct off
strip_query_terms on
coredump_dir none
redirector_bypass off
ignore_unknown_nameservers on
digest_generation on
digest_bits_per_entry 5
digest_rebuild_period 1 hour
digest_rewrite_period 1 hour
digest_swapout_chunk_size 4096 bytes
digest_rebuild_chunk_percentage 10
client_persistent_connections on
server_persistent_connections on
persistent_connection_after_error off
detect_broken_pconn off
balance_on_multiple_ip on
pipeline_prefetch off
request_entities off
high_response_time_warning 0
high_page_fault_warning 0
high_memory_warning 0
store_dir_select_algorithm least-load
ie_refresh off
vary_ignore_expire off
sleep_after_fork 0
minimum_expiry_time 60 seconds
relaxed_header_parser on
max_filedesc 1024
```

∞ LAMPIRAN 8 ∞

Isi dari *file* squidGuard.conf

```
# Haunz
# Copyright 2007
#
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~+>
# CONFIG FILE FOR SQUIDGUARD
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~+>
#
dbhome /var/squidGuard/cream
logdir /var/log/squid
#
# <+- Waktu -+>
#
time bebas {
            weekly    mtwhf 00:00 - 07:29
            weekly    mtwhf 15:31 - 24:00
            date      *.01.01                 # Tahun Baru Masehi
            date      *.08.17                 # Hari Kemerdekaan RI
            }
# <+- Sumber -+>
#
src lasif {
           ip 10.100.100.0/25                 # network lab
           }
#
# <+- Tujuan -+>
#
dest parno {
            domainlist                  parno-d
            urllist                     parno-u
            redirect                    302:http://172.17.63.129:8080
            }
#
dest multimedia {
                expressionlist          media
                redirect                302:http://172.17.63.129:8080
                }
#
acl {
      lasif within bebas {
      pass all
      }
      else {
            pass !in-addr !parno !multimedia all
            }
      default {
               pass                     none
               redirect                 http://www.google.com
               }
      }
```